



System Monitor Guide and Reference
Updated April, 2009

Note

Before using this information and the product it supports, read the general information under Appendix B, "Notices," on page 593.

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, 2009.

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
Setting monitor switches from a client application . . 5
Setting monitor switches from the CLP 7
Database system monitor data organization 8
Counter status and visibility 9
System monitor output: the self-describing data stream 10
Database system monitor memory requirements . . 11
Indoubt Transaction Manager overview 13
Monitoring with db2top in interactive mode commands. 16
.db2toprc configuration file 18

Chapter 2. System monitor switches . . 21

Chapter 3. Snapshot monitor 25

Access to system monitor data: SYSMON authority 25
Capturing database system snapshots using snapshot administrative views and table functions . 26
Capturing database system snapshot information to a file using the SNAP_WRITE_FILE stored procedure 28
Accessing database system snapshots using snapshot table functions in SQL queries (with file access) 30
Snapshot monitor SQL Administrative Views . . . 31
SQL access to database system snapshots 34
Capturing a database snapshot from the CLP . . . 35
Snapshot monitor CLP commands. 35
Capturing a database snapshot from a client application 38
Snapshot monitor API request types 39
Snapshot monitor sample output 41
Subsection snapshots 43
Global snapshots on partitioned database systems 44
Snapshot monitor self-describing data stream . . . 45

Chapter 4. Event monitors 49

Event types 49
Collecting information about database system events 51
Creating an event monitor 53
 Creating a table event monitor 53
 Event monitor table management 56
 Creating a file event monitor 60
 Event monitor file management 62
 Write-to-table and file event monitor buffering . 63
 Creating a pipe event monitor 64
 Event monitor named pipe management 65

 Creating an event monitor for partitioned databases 65
 Client identification within transaction and deadlock event monitors: Feature adoption reference 68
Event monitor sample output 68
 Formatting file or pipe event monitor output from a command line 75
 Event records and their corresponding applications 75
 Event monitor self-describing data stream . . . 76
 Transferring event monitor data between systems 78

Chapter 5. Activity Monitor overview 81

Monitoring scenarios 84
 Scenario: Identifying costly applications using snapshot administrative views 84
 Scenario: Monitoring buffer pool efficiency using administrative views 86
Setting up an activity monitor 87
Progress monitoring of the rollback process. . . . 87
Using snapshot monitor data to monitor the reorganization of a partitioned table 88
Inactive statement tracking for DEADLOCK WITH DETAILS HISTORY event monitors 95

Chapter 6. Working with the Memory Visualizer 97

Memory Visualizer overview 99

Chapter 7. Monitoring database systems (Windows). 103

Introduction to Windows Management Instrumentation (WMI) 103
DB2 database system integration with Windows Management Instrumentation 103
Windows performance monitor introduction . . . 105
 Registering DB2 with the Windows performance monitor 105
 Enabling remote access to DB2 performance information 106
 Displaying DB2 database and DB2 Connect performance values 106
 Windows performance objects. 106
 Accessing remote DB2 database performance information 107
 Resetting DB2 performance values 108

Part 2. System monitor elements 109

Chapter 8. Logical data groups 111

Snapshot monitor interface mappings to logical data groups 111

Snapshot monitor logical data groups and monitor elements	115
Event type mappings to logical data groups	145
Event monitor logical data groups and monitor elements	148
Logical data groups affected by COLLECT ACTIVITY DATA settings	169

Chapter 9. Database system monitor elements 171

Server identification and status monitor elements	172
db2start_time - Start Database Manager Timestamp	172
server_instance_name - Server Instance Name	172
server_db2_type - Database Manager Type at Monitored (Server) Node	173
server_prdid - Server Product/Version ID	173
server_version - Server Version	174
service_level - Service Level	174
server_platform - Server Operating System	175
product_name - Product Name	175
db2_status - Status of DB2 Instance	175
time_zone_disp - Time Zone Displacement	176
Database identification and status monitor elements	176
db_name - Database Name	176
db_path - Database Path.	177
db_conn_time - Database Activation Timestamp	178
conn_time - Time of Database Connection	178
disconn_time - Database Deactivation Timestamp	178
db_status - Status of Database.	179
catalog_node_name - Catalog Node Network Name	179
db_location - Database Location	180
catalog_node - Catalog Node Number	180
last_backup - Last Backup Timestamp	180
db_storage_path - Automatic storage path.	181
num_db_storage_paths - Number of automatic storage paths	181
sto_path_free_sz - Automatic Storage Path Free Space	181
fs_used_size - Amount of Space Used on a File System	182
fs_total_size - Total Size of a File System	182
fs_id - Unique File System Identification Number	183
fs_type - File System Type	183
Application identification and status monitor elements	184
agent_id - Application Handle (agent ID)	184
appl_status - Application Status	185
codepage_id - ID of Code Page Used by Application	187
status_change_time - Application Status Change Time	187
appl_id_oldest_xact - Application with Oldest Transaction	188
smallest_log_avail_node - Node with Least Available Log Space	188
appl_name - Application Name	189

appl_id - Application ID.	189
sequence_no - Sequence number monitor element	191
auth_id - Authorization ID	192
session_auth_id - Session Authorization ID	193
client_prdid - Client Product/Version ID	193
client_db_alias - Database Alias Used by Application	194
host_prdid - Host Product/Version ID	194
is_system_appl - Is System Application monitor element	195
outbound_appl_id - Outbound Application ID	195
outbound_sequence_no - Outbound Sequence Number	196
execution_id - User Login ID	196
corr_token - DRDA Correlation Token	197
client_pid - Client Process ID	197
client_platform - Client Operating Platform	198
client_protocol - Client Communication Protocol	198
territory_code - Database Territory Code	199
appl_priority - Application Agent Priority.	199
appl_priority_type - Application Priority Type	200
authority_lvl - User Authorization Level	200
authority_bitmap - User Authorization Level monitor element	201
node_number - Node Number	202
coord_node - Coordinating Node.	203
appl_con_time - Connection Request Start Timestamp	203
connections_top - Maximum Number of Concurrent Connections.	204
conn_complete_time - Connection Request Completion Timestamp	204
prev_uow_stop_time - Previous Unit of Work Completion Timestamp	205
uow_start_time - Unit of Work Start Timestamp	205
uow_stop_time - Unit of Work Stop Timestamp	206
uow_elapsed_time - Most Recent Unit of Work Elapsed Time	206
uow_comp_status - Unit of Work Completion Status	207
uow_status - Unit of Work Status.	207
appl_idle_time - Application Idle Time	208
DB2 agent information monitor elements	208
Database manager configuration monitor elements	209
Agents and connections monitor elements.	209
Memory pool monitor elements	220
Sort monitor elements	224
Hash join monitor elements	231
On-Line Analytical Processing (OLAP) monitor elements	234
Fast communications manager (FCM) monitor elements	237
Database configuration monitor elements	239
Buffer pool activity monitor elements	239
Non-buffered I/O activity monitor elements	272
Catalog cache monitor elements	276
Package cache monitor elements	280
SQL workspaces monitor elements	283
Database heap monitor elements	290
Logging monitor elements	290

Database and application activity monitor elements	301	dcx_db_name - DCS Database Name	434
blocks_pending_cleanup - Pending cleanup		host_db_name - Host Database Name	434
rolled-out blocks monitor element	301	gw_db_alias - Database Alias at the Gateway	434
Locks and deadlocks monitor elements	301	gw_con_time - DB2 Connect Gateway First	
Lock wait information monitor elements	317	Connect Initiated	435
Rollforward monitoring monitor elements	323	gw_connections_top - Maximum Number of	
Table space activity monitor elements	324	Concurrent Connections to Host Database	435
Table activity monitor elements	346	gw_total_cons - Total Number of Attempted	
Table reorganization monitor elements	359	Connections for DB2 Connect	435
SQL cursors monitor elements	364	gw_cur_cons - Current Number of Connections	
SQL and XQuery statement activity monitor		for DB2 Connect	436
elements	367	gw_cons_wait_host - Number of Connections	
SQL statement details monitor elements	377	Waiting for the Host to Reply	436
Subsection details monitor elements	396	gw_cons_wait_client - Number of Connections	
Dynamic SQL monitor elements	401	Waiting for the Client to Send Request	437
Intra-query parallelism monitor elements	403	gw_exec_time - Elapsed Time Spent on DB2	
CPU usage monitor elements	404	Connect Gateway Processing	437
Snapshot monitoring monitor elements	408	sql_stmts - Number of SQL Statements	
Event monitoring monitor elements	410	Attempted	437
Utilities monitor elements	416	sql_chains - Number of SQL Chains Attempted	438
High availability disaster recovery (HADR)		open_cursors - Number of Open Cursors	439
monitor elements	422	dcx_appl_status - DCS Application Status	439
hadr_role - HADR Role	422	agent_status - DCS Application Agents	440
hadr_state - HADR State monitor element	423	host_ccsid - Host Coded Character Set ID	440
hadr_syncmode - HADR Synchronization Mode		outbound_comm_protocol - Outbound	
monitor element	424	Communication Protocol	441
hadr_connect_status - HADR Connection Status		outbound_comm_address - Outbound	
monitor element	424	Communication Address	441
hadr_connect_time - HADR Connection Time		inbound_comm_address - Inbound	
monitor element	425	Communication Address	441
hadr_heartbeat - HADR Heartbeat monitor		inbound_bytes_received - Inbound Number of	
element	426	Bytes Received	442
hadr_local_host - HADR Local Host monitor		outbound_bytes_sent - Outbound Number of	
element	427	Bytes Sent	442
hadr_local_service - HADR Local Service		outbound_bytes_received - Outbound Number	
monitor element	427	of Bytes Received	442
hadr_remote_host - HADR Remote Host		inbound_bytes_sent - Inbound Number of Bytes	
monitor element	428	Sent	443
hadr_remote_service - HADR Remote Service		outbound_bytes_sent_top - Maximum	
monitor element	428	Outbound Number of Bytes Sent	443
hadr_remote_instance - HADR Remote Instance		outbound_bytes_received_top - Maximum	
monitor element	429	Outbound Number of Bytes Received	444
hadr_timeout - HADR Timeout monitor element	429	outbound_bytes_sent_bottom - Minimum	
hadr_primary_log_file - HADR Primary Log File		Outbound Number of Bytes Sent	444
monitor element	429	outbound_bytes_received_bottom - Minimum	
hadr_primary_log_page - HADR Primary Log		Outbound Number of Bytes Received	445
Page monitor element	430	max_data_sent_128 - Number of Statements	
hadr_primary_log_lsn - HADR Primary Log		with Outbound Bytes Sent Between 1 and 128	
LSN monitor element	430	Bytes	445
hadr_standby_log_file - HADR Standby Log File		max_data_received_128 - Number of Statements	
monitor element	431	with Outbound Bytes Received Between 1 and	
hadr_standby_log_page - HADR Standby Log		128 Bytes	445
Page monitor element	431	max_data_sent_256 - Number of Statements	
hadr_standby_log_lsn - HADR Standby Log		with Outbound Bytes Sent Between 129 and 256	
LSN monitor element	432	Bytes	446
hadr_log_gap - HADR Log Gap	432	max_data_received_256 - Number of Statements	
hadr_peer_window - HADR peer window		with Outbound Bytes Received Between 129	
monitor element	433	and 256 Bytes	446
hadr_peer_window_end - HADR peer window		max_data_sent_512 - Number of Statements	
end monitor element	433	with Outbound Bytes Sent Between 257 and 512	
DB2 Connect monitor elements	434	Bytes	447

max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes	447	max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms	456
max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes	447	max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms	456
max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes	448	max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms	457
max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes	448	network_time_top - Maximum Network Time for Statement	457
max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes	449	network_time_bottom - Minimum Network Time for Statement	457
max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes	449	xid - Transaction ID	458
max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes	450	elapsed_exec_time - Statement Execution Elapsed Time	458
max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes	450	host_response_time - Host Response Time.	459
max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes	450	num_transmissions - Number of Transmissions	460
max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes	451	num_transmissions_group - Number of Transmissions Group	460
max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes.	451	con_response_time - Most Recent Response Time for Connect	461
max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes	452	con_elapsed_time - Most Recent Connection Elapsed Time	461
max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element	452	gw_comm_errors - Communication Errors.	461
max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes	453	gw_comm_error_time - Communication Error Time	462
max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element	453	blocking_cursor - Blocking Cursor	462
max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes	453	transaction_processor_monitoring_monitor_elements	462
max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes	454	Federated database systems monitor elements	464
max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms	454	datasource_name - Data Source Name	465
max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms	455	disconnects - Disconnects	465
max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms	455	insert_sql_stmts - Inserts	465
		update_sql_stmts - Updates	466
		delete_sql_stmts - Deletes	467
		create_nickname - Create Nicknames	467
		passthru - Pass-Through	468
		stored_procs - Stored Procedures	468
		remote_locks - Remote Locks	468
		sp_rows_selected - Rows Returned by Stored Procedures	469
		select_time - Query Response Time	469
		insert_time - Insert Response Time	470
		update_time - Update Response Time	471
		delete_time - Delete Response Time	471
		create_nickname_time - Create Nickname Response Time	472
		passthru_time - Pass-Through Time	472
		stored_proc_time - Stored Procedure Time.	473
		remote_lock_time - Remote Lock Time	473
		Workload management monitor elements	474
		activate_timestamp - Activate timestamp monitor element	474
		activity_collected - Activity collected monitor element	474
		activity_id - Activity ID monitor element	475
		activity_secondary_id - Activity secondary ID monitor element	475
		activity_type - Activity type monitor element	476

act_exec_time - Activity execution time monitor element	476
act_total - Activities total monitor element.	477
arm_correlator - Application response measurement correlator monitor element	477
bin_id - Histogram bin identifier monitor element	478
bottom - Histogram bin bottom monitor element	478
concurrent_act_top - Concurrent activity top monitor element	478
concurrent_connection_top - Concurrent connection top monitor element	479
concurrent_wlo_act_top - Concurrent WLO activity top monitor element	479
concurrent_wlo_top - Concurrent workload occurrences top monitor element	480
coord_act_aborted_total - Coordinator activities aborted total monitor element	480
coord_act_completed_total - Coordinator activities completed total monitor element.	481
coord_act_lifetime_top - Coordinator activity lifetime top monitor element	481
coord_act_rejected_total - Coordinator activities rejected total monitor element	481
coord_partition_num - Coordinator partition number monitor element	482
cost_estimate_top - Cost estimate top monitor element	482
coord_act_lifetime_avg - Coordinator activity lifetime average monitor element.	483
coord_act_queue_time_avg - Coordinator activity queue time average monitor element.	483
coord_act_exec_time_avg - Coordinator activities execution time average monitor element	484
request_exec_time_avg - Request execution time average monitor element	485
coord_act_est_cost_avg - Coordinator activity estimated cost average monitor element	485
coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element	486
db_work_action_set_id - Database work action set ID monitor element	487
db_work_class_id - Database work class ID monitor element	487
histogram_type - Histogram type monitor element	488
last_wlm_reset - Time of last reset monitor element	489
num_threshold_violations - Number of threshold violations monitor element	489
number_in_bin - Number in bin monitor element	489
parent_activity_id - Parent activity ID monitor element	490
parent_uow_id - Parent unit of work ID monitor element	490
prep_time - Preparation time monitor element	491
queue_assignments_total - Queue assignments total monitor element.	491

queue_size_top - Queue size top monitor element	491
queue_time_total - Queue time total monitor element	492
rows_fetched - Rows fetched monitor element	492
rows_modified - Rows modified monitor element	492
rows_returned - Rows returned monitor element	493
rows_returned_top - Actual rows returned top monitor element	493
sc_work_action_set_id - Service class work action set ID monitor element	494
sc_work_class_id - Service class work class ID monitor element	494
section_env - Section environment monitor element	495
service_class_id - Service class ID monitor element	495
service_subclass_name - Service subclass name monitor element	495
service_superclass_name - Service superclass name monitor element	496
statistics_timestamp - Statistics timestamp monitor element	496
temp_tablespace_top - Temporary table space top monitor element	497
threshold_action - Threshold action monitor element	497
threshold_domain - Threshold domain monitor element	498
threshold_maxvalue - Threshold maximum value monitor element	498
threshold_name - Threshold name monitor element	499
threshold_predicate - Threshold predicate monitor element	499
threshold_queuesize - Threshold queue size monitor element	499
thresholdid - Threshold ID monitor element	500
time_completed - Time completed monitor element	500
time_created - Time created monitor element	501
time_of_violation - Time of violation monitor element	501
time_started - Time started monitor element	501
top - Histogram bin top monitor element	502
uow_id - Unit of work ID monitor element	502
wlo_completed_total - Workload occurrences completed total monitor element	502
work_action_set_id - Work action set ID monitor element	503
work_action_set_name - Work action set name monitor element	503
work_class_id - Work class ID monitor element	504
work_class_name - Work class name monitor element	504
workload_id - Workload ID monitor element	504
workload_name - Workload name monitor element	505
workload_occurrence_id - Workload occurrence identifier monitor element	505

Real-time statistics monitor elements	506
stats_cache_size – Size of statistics cache monitor element	506
stats_fabrications – Total number of statistics fabrications monitor elements	506
sync_runstats – Total number of synchronous RUNSTATS activities monitor element	507
async_runstats – Total number of asynchronous RUNSTATS requests monitor element	507
stats_fabricate_time – Total time spent on statistics fabrication activities monitor element	508
sync_runstats_time – Total time spent on synchronous RUNSTATS activities monitor element	509

Chapter 10. Database system monitor interfaces 511

Part 3. Monitoring database health 513

Chapter 11. Introduction to the health monitor 515

Health indicators	515
Health indicator process cycle	517
Enabling health alert notification	518

Chapter 12. Health Center overview 521

Investigating alert conditions	523
--	-----

Chapter 13. Health monitor 525

Health indicator data	526
Capturing database health snapshots	526
Capturing a database health snapshot using SQL table functions	526
Capturing a database health snapshot using the CLP	527
Capturing a database health snapshot from a client application	528
Health monitor sample output	530
Global health snapshots	532
Graphical tools for the health monitor	533
Retrieving health recommendations	535
Health recommendation queries with SQL.	535
Retrieving health recommendations using the CLP	535
Retrieving health recommendations using a client application	539
Resolving health monitor alerts using the Health Center.	540
Health indicator configuration.	541
Retrieving health indicator configuration using the CLP	543
Health indicator configuration updates using the CLP	543
Resetting health indicator configuration using the CLP	544
Configuring health indicators using a client application	545

Configuring health indicators using the Health Center.	547
Health monitor alert actions on combined states	548

Part 4. Health indicators 551

Chapter 14. Health monitor interface mappings to logical data groups . . . 553

Chapter 15. Health indicators summary 555

Health indicator format	557
Table space storage health indicators	558
Health indicators for DMS table spaces.	558
db.auto_storage_util - Database automatic storage utilization health indicator	558
ts.ts_auto_resize_status - Table space automatic resize status health indicator	559
ts.ts_util_auto_resize - Automatic resize table space utilization health indicator	560
ts.ts_util - Table Space Utilization.	560
tsc.tscont_util - Table Space Container Utilization	561
ts.ts_op_status - Table Space Operational State	562
tsc.tscont_op_status - Table Space Container Operational State	562
Sorting health indicators.	562
db2.sort_privmem_util - Private Sort Memory Utilization	562
db.sort_shrmem_util - Shared Sort Memory Utilization	563
db.spilled_sorts - Percentage of Sorts That Overflowed	564
db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization.	564
Database manager (DBMS) health indicators	565
db2.db2_op_status - Instance Operational State	565
Instance Highest Severity Alert State	565
Database health indicators	566
db.db_op_status - Database Operational State	566
Database Highest Severity Alert State	566
Maintenance health indicators.	567
db.tb_reorg_req - Reorganization Required	567
db.tb_runstats_req - Statistics Collection Required	567
db.db_backup_req - Database Backup Required	568
High availability disaster recovery (HADR) health indicators.	568
db.hadr_op_status - HADR Operational Status	568
db.hadr_delay - HADR Log Delay	569
Logging health indicators	569
db.log_util - Log Utilization	569
db.log_fs_util - Log File System Utilization	569
Application concurrency health indicators.	570
db.deadlock_rate - Deadlock Rate	570
db.locklist_util - Lock List Utilization	571
db.lock_escal_rate - Lock Escalation Rate	571
db.apps_waiting_locks - Percentage of Applications Waiting on Locks	572

Package cache, catalog cache, and workspace health indicators	573
db.catcache_hitratio - Catalog Cache Hit Ratio	573
db.pkgcache_hitratio - Package Cache Hit Ratio	573
db.shrworkspace_hitratio - Shared Workspace Hit Ratio	573
Memory health indicators	574
db2.mon_heap_util - Monitor Heap Utilization	574
db.db_heap_util - Database Heap Utilization	574
Federated health indicators	575
db.fed_nicknames_op_status - Nickname Status	575
db.fed_servers_op_status - Data Source Server Status	575
Chapter 16. Health monitor interfaces	577
Health monitor SQL table functions	578
Health monitor CLP commands	578
Health monitor API request types	579

Part 5. Appendixes 581

Appendix A. Overview of the DB2 technical information	583
DB2 technical library in hardcopy or PDF format	583
Ordering printed DB2 books	586
Displaying SQL state help from the command line processor	587
Accessing different versions of the DB2 Information Center	587
Displaying topics in your preferred language in the DB2 Information Center	587
Updating the DB2 Information Center installed on your computer or intranet server	588
DB2 tutorials	590
DB2 troubleshooting information	590
Terms and Conditions	590
Appendix B. Notices	593
Index	597

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.

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.

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, SYSMMAINT, 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.

---

## 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:

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

### 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” on page 11.
- 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<sup>®</sup>, 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<sup>®</sup> 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<sup>®</sup> 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.

---

## Monitoring with db2top in interactive mode commands

The db2top monitoring utility quickly and efficiently monitors a complex DB2 environment. It combines DB2 snapshot information from all database partitions and provides a dynamic, real-time view of a running DB2 system using a text-based user interface.

When you run db2top in interactive mode, you can issue the following commands:

- A** Monitor either the primary or the secondary database in a HADR cluster.
- a** Goto application details for agent (or restrict on agent on statement screen). The db2top command will prompt for the agent-id.
- B** Display the main consumer of critical server resources (Bottleneck Analysis).
- c** This option allows to change the order of the columns displayed on the screen. The syntax is in the form: 1,2,3,... where 1,2,3 correspond respectively to the 1st, 2nd and 3rd columns displayed. These are the column numbers to use when specifying a sort criteria.

When you use the c switch key, a screen is shown specifying the order of the columns displayed on screen. The left part of the screen displays the default order and column numbers; the right part of the screen displays the current ordering. To change the order of the columns, enter the new column order in the text field at the bottom of the screen. Next, enter the relative column positions as displayed on the left, separated by commas. Not all columns need to be specified. This column ordering can be saved in `$DB2TOPRC` for subsequent db2top monitoring sessions by selecting w. You can sort and select in which order the columns are displayed on the screen. Valid keywords for column ordering in the `.db2toprc` file are:

- sessions=
  - tables=
  - tablespaces=
  - bufferpools=
  - dynsql=
  - statements=
  - locks=
  - utilities=
  - federation=
- b** Goto buffer pool screen.
  - C** Toggle snapshot data collector on/off.
  - d** Goto database screen.
  - D** Goto the dynamic SQL screen.

- f** Freeze screen.
- F** Monitor federated queries on the primary server.
- G** Toggle graph on/off.
- h** Go to Help screen
- H** Goto the history screen
- i** Toggle idle sessions on/off.
- k** Toggle actual vs delta values.
- l** Goto sessions screen.
- L** Allows to display the complete query text from the SQL screen. Regular DB2 explain can then be run using e or X options.
- m** Display memory pools.
- o** Display session setup.
- p** Goto the partitions screen.
- P** Select db partition on which to issue snapshot.
- q** Quit db2top.
- R** Reset snapshot data.
- s** Goto the statements screen.
- S** Run native DB2 snapshot.
- t** Goto table spaces screen.
- T** Goto tables screen
- u** Display active utilities and aggregate them across database partitions.
- U** Goto the locks screen.
- V** Set default explains schema.
- w** Write session settings to .db2toprc.
- W** Watch mode for agent\_id, os\_user, db\_user, application or netname. Statements returned by the session snapshot (option l) will be written to agent.sql, os\_user-agent.sql, db\_user-agent.sql, application-agent.sql or netname-agent.sql. When issued from the dynamic SQL screen (option D), statements will be written to db2adv.sql in a format compatible with db2advvis.
- X** Toggle extended mode on/off.
- z|Z** Sort on ascending or descending order.
- /** Enter expression to filter data. Expression must conform to regular expression. Each function (screen) can be filtered differently. The regexp check is applied to the whole row.
- <l>** Move to left or right of screen.

The following switches apply to the applications screen only:

- r** Return to previous function.
- R** Toggle automatic refresh.
- g** Toggle graph on/off.

- X Toggle extended mode on/off.
- d Display agents.

To start db2top in interactive mode, issue the following command:

```
db2top -d <database name>
```

When you type

```
db2top -d sample
```

the following output is displayed:

```
[\]11:57:10,refresh=2secs(0.000) Inactive,part=[1/1],<instanceName>:sample
[d=Y,a=N,e=N,p=ALL] [qp=off]
```

[/]: When rotating, it means that db2top is waiting between two snapshots, otherwise, it means db2top is waiting from an answer from DB2

11:57:10: current time

refresh=2secs: time interval

refresh=!secs: Exclamation mark means the time to process the snapshot by DB2 is longer than the refresh interval. In this case, db2top will increase the interval by 50%.

If this occurs too often because the system is too busy, you can either increase the snapshot interval (option I), monitor a single database partition (option P), or turn off extended display mode (option x)

0.000 : time spent inside DB2 to process the snapshot

d=Y/N : delta or cumulative snapshot indicator (command option -k or option k).

a=Y/N : Active only or all objects indicator (-a command option set or i)

e=Y/N : Extended display indicator

p=ALL : All database partitions

p=CUR: Current database partition (-P command option with no partition number specified)

p=3 : target database partition number: say 3

Inactive: : Shows inactive if DB2 is not running, otherwise displays the platform on which DB2 is running

part=[1/1] : active database partition number vs total database partition number.

For example, part=[2,3] means one database partition out of 3 is down (2 active, 3 total)

<instanceName> : instance name

sample : database name

qp=off/on : query patroller indicator (DYNMGM database configuration parameter)

for the database partition on which db2top is attached

The following example demonstrates running the db2top monitoring utility in interactive mode in a partitioned database environment:

```
db2top -d TEST -n mynode -u user -p passwd -V skm4 -B -i 1
```

The command parameters are as follows:

```
-d TEST # database name
-n mynode # node name
-u user # user id
-p passwd # password
-V skm4 # Schema name
-B # Bold enabled
-i 1 # Screen update interval: 1 second
```

---

## .db2toprc configuration file

The .db2toprc configuration file is a user generated file used to set parameters at initialization time for the db2top monitoring utility.

The db2top utility will search for the location of the .db2toprc file using the user-defined variable *\$db2topRC*. If the variable is not set, db2top will first search for the .db2toprc file in the current directory and then in the home directory. The .db2toprc file is user generated.



## Environment variables

You can set the following environment variables:

- **DB2TOPRC**

A user defined environment variable that stores the location of the .db2toprc file. For example, on Linux®, you can define **DB2TOPRC** as: `export db2toprc=~/.db2toprc`.

If the variable is not set by the user, db2top will first search for the .db2toprc file in the current directory and then in the home directory.

- **DB2DBDFT**

This variable specifies the database alias name of the database to be used for implicit connects. It is used when no database name is specified on the command line or in the .db2toprc configuration file.

- **EDITOR**

This system environment variable specifies the command used to start the text editor used to display the results of explain or native snapshots.

If this variable is not set, vi is used.

## Structure

Some of the entries in the .db2toprc file are described here.

### **cpu=command**

Use this entry to display the results of CPU activity on the second line at the right of the screen output. For example:

```
cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)0,$14+$15);}'
```

displays Cpu=2(usr+sys) on the right of the screen.

### **io=command**

Use this entry to specify a command and display the result on the second line at the left of the screen output. For example:

```
io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)0,$10+$11);}'
```

displays Disk=76(bi+bo) on the left of the screen.

Both commands run as background processes and the fields on the screen are updated asynchronously.

### **shell alias=command**

Use this shell entry to specify a user defined command, for example: `shell M=top` spawns top from a db2top session when entering M. It returns to the current screen upon exit.

### **function alias=command**

Use this entry to specify a user defined command, for example: `function N=netstat` creates a new function called N that repeatedly displays the output of netstat. There can be multiple function entries. They must be placed on separate lines. For example:

```
function Q=netstat
function N=df -k
```

### **sort=command**

Use this entry to specify a sort order, for example: `sort=command` creates a default sort order for this function, where command is the column number. It can be either ascending or descending. Sort is valid for sessions, tables, tablespace, bufferpool, dynsql, statements, locks, utilities and federation.

## Sample .db2toprc file

There is no default .db2toprc configuration file. However, you can press "W" to create a .db2toprc for the current setup. Use the following sample .db2toprc file as a reference. Comments have been added to all entries.

```
db2top configuration file
On unix, should be located in $HOME/<cmdname> .db2toprc</cmdname>
File generated by db2top-1.0a
#
node= # [-n] nodename
database=sample # [-d] databasename
user= # [-u] database user
password= # [-p] user password (crypted)
schema= # [-V] default schema for explains
interval=2 # [-i] sampling interval
active=OFF # [-a] display active sessions only (on/off)
reset=OFF # [-R] Reset snapshot at startup (on/off)
delta=ON # [-k] Toggle display of delta/cumulative values (on/off)
gauge=ON # display graph on sessions list (on/off)
colors=ON
True if terminal supports colors. Informs GE_WRS if it can display information with colors
graphic=ON # True if terminal supports semi graphical characters (on/off).
port= # Port for network collection
streamsize=size # Max collection size per hour (eg. 1024 or 1K : K, M or G)
Command to get cpu usage information from OS
cpu=vmstat 2 2 | tail -1 | awk '{printf("%d(usr+sys)0,$14+$15);}'
Command to get IO usage information from OS
io=vmstat 2 2 | tail -1 | awk '{printf("%d(bi+bo)0,$10+$11);}'
Ordering of information in sessions screen
Column order for the session screen (option l)
sessions=0,1,18,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,19,20,21,22,23
Column order for the tables screen (option T)
tables=0,1,2,4,3,5,6,7
Column order for the tablespaces screen (option t).
The display will be sorted in ascending order on column #22
tablespaces=0,1,18,2,3,4,5,6,7,8, sort=22a
Column order for the bufferpool screen (option b)
bufferpools=0,1,18,2,3,4,5,6,7,8,9,10
Column order for the Dynamic SQL screen (option D)
dynsql=0,1,18,2,3,4,5,6,7,8,9
statements=0,1
locks=0,1
utilities=0 # contains the default column and sort order for the utility screen
federation=0,2,4 # contains the default column and sort order for the federation screen

User defined commands
shell P=top
function N=date && netstat -t tcp
```

---

## 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.



---

## 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 28, 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, SYSMANT, 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 <b>was accessed</b> by an application connected to the database. Requires the table switch. |
| Lock          | get snapshot for locks for application applid <i>appl-id</i>      | List of locks held by the application. Lock wait information requires the lock switch.                                                                                                                                                                                         |
| Lock          | get snapshot for locks for application agentid <i>appl-handle</i> | List of locks held by the application. Lock wait information requires the lock switch.                                                                                                                                                                                         |
| Lock          | get snapshot for locks on <i>dbname</i>                           | Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.                                                                                                                                        |
| Table space   | get snapshot for tablespaces on <i>dbname</i>                     | Information about table space activity for a database. Requires the buffer pool switch. Also included is information 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 was accessed 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 1 shows the structure of the data stream and Table 8 on page 46 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.

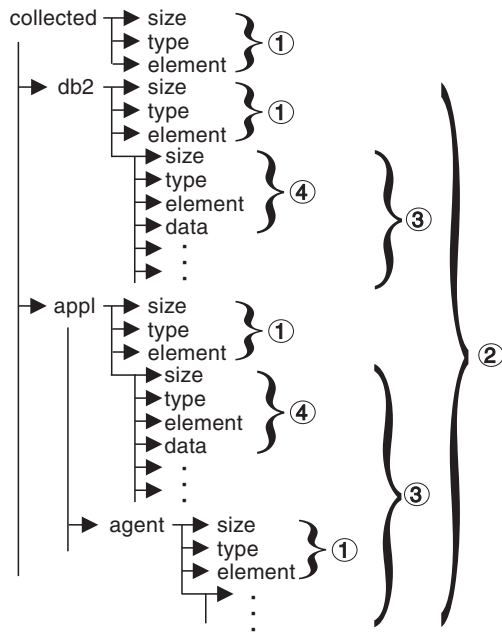


Figure 1. Snapshot Monitor Data Stream

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. Note: 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®, 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.

- *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.

Note that if an activity event monitor is active when the database deactivates, any backlogged activity records in the queue are discarded. To ensure that you obtain all activity event monitor records and that none are discarded, explicitly deactivate the activity event monitor first before deactivating the database.

When an activity event monitor is explicitly deactivated, all backlogged activity records in the queue are processed before the event monitor deactivates.

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 dlmon 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
```

- 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.

- 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))
BUFFERSIZE 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.

- 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))
BUFFERSIZE 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.

- 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
BUFFERSIZE 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.

- 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 TABLE BUFFERSIZE 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.

Whenever a write-to-table event monitor activates, it will acquire IN table locks on each target table in order to prevent them from being modified while the event monitor is active. Table locks are maintained on all tables while the event monitor is active. If exclusive access is required on any of the target tables (for example, when a utility is to be run), first deactivate the event monitor to release the table locks before attempting such access.

Each column name in a target table matches an event monitor element identifier. Any event monitor element that does not have a corresponding target table column is ignored.

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.

For SMS table spaces, the value is set to 100. It is recommended that when the target table space has auto-resize enabled the PCTDEACTIVATE be set to 100.

In a non-partitioned database environment, all write to table event monitors are deactivated when the last application terminates (and the database has not been explicitly activated). In a partitioned database environment, write to table event monitors are deactivated when the catalog partition deactivates.

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
	CONTROL	Event monitor metadata

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

Event type	Target table names	Available information
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
	CONNMEMUSE	Memory pool metadata
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

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

Event type	Target table names	Available information
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
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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/d1events' 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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/d1events' BUFFERSIZE 8
NONBLOCKED MAXFILES NONE
```

- 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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
WRITE TO FILE '/tmp/d1events' 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
```

- 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
```

- 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.

Client identification within transaction and deadlock event monitors: Feature adoption reference

The addition of the *ClientUserID*, *ClientWrkstnName*, *ClientApplName*, and *ClientAcctng* elements in DB2 V9.5 Fix Pack 3 allows you to collect more detail than ever from your transaction and deadlock event monitors. Transaction and deadlock event monitors created prior to using Fix Pack 3 will operate without any issue, however, if you wish to leverage the new elements added in Fix Pack 3 you will need to update your event monitors.

Options for updating your transaction event monitors

There are two options available to you in order to include the new elements added in Fix Pack 3. You can either: drop and recreate the event monitor, or alter the event monitor.

Scenario 1: Alter the existing event monitor target table.

Assuming you have an active "WRITE TO TABLE" event monitor named FOO_FP2, and a target table named XACT_FOO_FP2, you may use:

```
db2 "SET EVENT MONITOR FOO_FP2 STATE 0"
db2 "ALTER TABLE XACT_FOO_FP2
  ADD COLUMN TPMON_ACC_STR VARCHAR(200)
  ADD COLUMN TPMON_CLIENT_APP VARCHAR(256)
  ADD COLUMN TPMON_CLIENT_USERID VARCHAR(256)
  ADD COLUMN TPMON_CLIENT_WKSTN VARCHAR(256)"
db2 "REORG TABLE XACT_FOO_FP2"
db2 "SET EVENT MONITOR FOO_FP2 STATE 1"
```

Scenario 2: Drop the existing event monitor target table and create a new event monitor and target table.

Again assuming you have an active "WRITE TO TABLE" event monitor named FOO_FP2, and a target table named XACT_FOO_FP2, you may use:

```
db2 "SET EVENT MONITOR FOO_FP2 STATE 0"
db2 "DROP EVENT MONITOR FOO_FP2"
db2 "DROP TABLE XACT_FOO_FP2"
db2 "CREATE EVENT MONITOR FOO_FP2 FOR TRANSACTIONS WRITE TO TABLE"
db2 "SET EVENT MONITOR FOO_FP2 STATE 1"
```

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/db2samp1`
- Windows: `sqllib\bin\db2samp1.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 : 0x02000F00000000000000000000000054
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 : 0x020005000000000000000000000054
 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
- SYSTRL
- SYSMANT
- 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 2 shows the structure of the data stream and Table 12 on page 77 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.

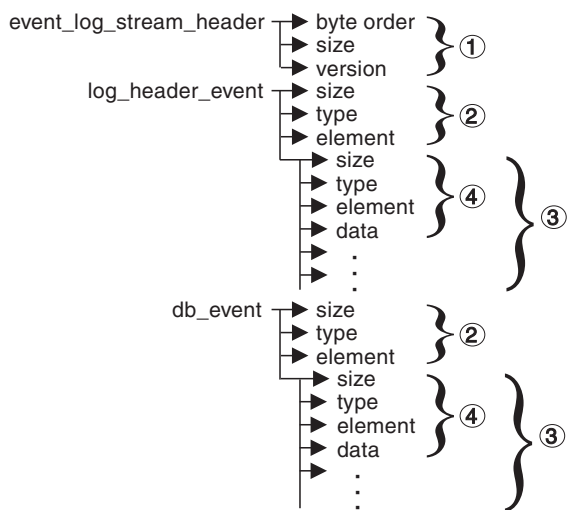


Figure 2. Event Monitor Data Stream

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.
- 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.
 - The size element in the header indicates the size of all the data in that logical data group.
 - 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 little_endian	The name of the monitor element collected. 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 850	The name of the monitor element collected. 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 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 */

```

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(1) (((1) >> 24) & 0xFF) | (((1) & 0xFF0000) >> 8) & 0xFF00 \
    | (((1) & 0xFF00) << 8) | ((1) << 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 */

```

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 3 on page 82 describes the process for using the Activity monitor to solve a problem.

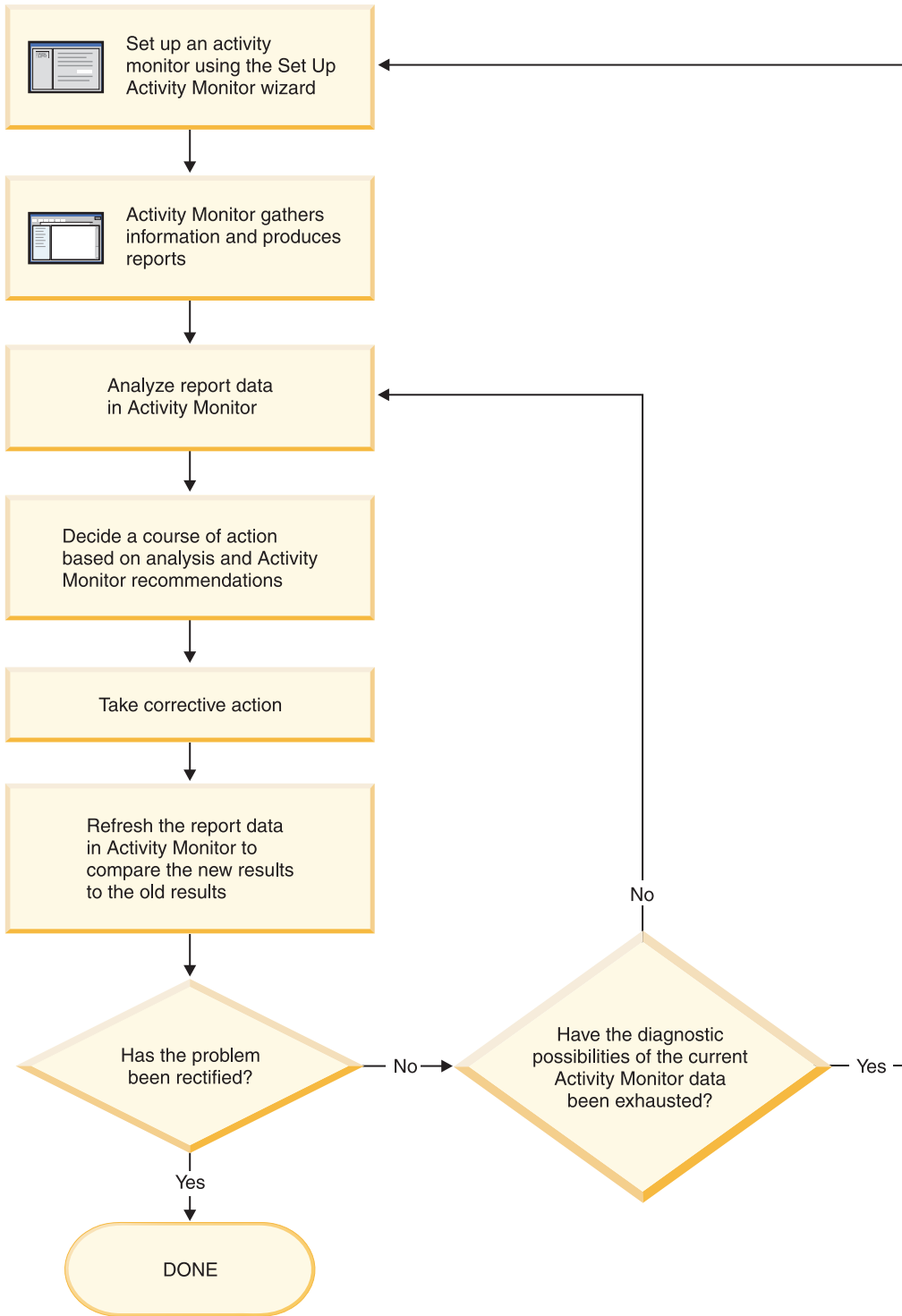


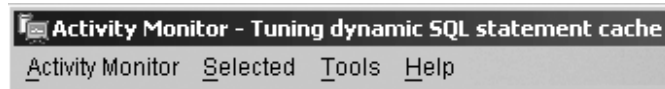
Figure 3. Activity Monitor overview

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 Report data pane. Right-click and select Show Latest Transactions . The Application Transactions window opens.
Statements	View SQL statements running on a selected application.	Select one or more applications in the Report data pane. Right-click and select Show Latest Statements . 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 Report data pane. Select Show Statement Text .
Application Lock Chains	View locks and lock-waiting situations that currently affect a selected application.	Select an application in the Report data pane. Right-click and select Show Lock Chains . 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 About .
	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 Show Lock Details .
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 Report arrow to select the report and click the Report Details push button. View the Details page.
	View recommendations provided by Activity Monitor	From an Activity Monitor window, an Application Statements window, or an Application Transactions window, use the Report arrow to select the report and click the Report Details push button. View the Recommendations 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 rollback process

If you obtain an application snapshot while a transaction is rolling back, you will see rollback monitor elements in the output. This information can be used to monitor the progress of the rollback operation.

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

The following is an example of output from the 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
```

The value in the application status monitor element implies which type of rollback event is occurring:

Rollback Active

This is a unit of work rollback: an explicit (user invoked) or implicit (forced) rollback of the entire transaction.

Rollback to Savepoint

This is a partial rollback to a statement or application level savepoint. Nested savepoints are considered a single unit, using the outermost 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.

Total Work units is an estimate based on the range of log records in the log stream that need to be rolled back for the transaction or savepoint. It does not indicate the exact number of log record bytes that need to be processed.

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 97.

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 [show detail]	applications	appl_info
SQLMA_DBASE_APPLINFO	list applications for database <i>dbname</i> [show detail]	applications	appl_info
SQLMA_DCS_APPLINFO_ALL	list dcs applications [show detail]		dcs_appl_info
SQLMA_DB2	get snapshot for dbm	SNAPDBM	db2
		SNAPFCM	fcm
		SNAPFCMPART	fcm_node
		SNAPUTIL	utility_info
		SNAPUTIL_PROGRESS	progress, progress_info
		SNAPDBM_MEMORY_POOL	memory_pool
	get dbm monitor 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	

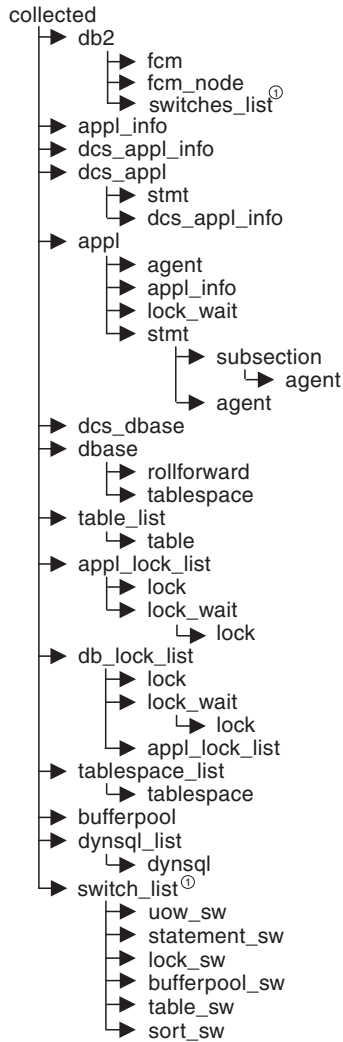
Table 14. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

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 4. 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 208 “lock_timeout_val - Lock timeout” on page 320

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 366
	"agent_sys_cpu_time - System CPU Time used by Agent" on page 404
	"agent_usr_cpu_time - User CPU Time used by Agent" on page 404
	"agents_stolen - Stolen Agents" on page 218
	"appl_con_time - Connection Request Start Timestamp" on page 203
	"appl_idle_time - Application Idle Time" on page 208
	"appl_priority - Application Agent Priority" on page 199
	"appl_priority_type - Application Priority Type" on page 200
	"associated_agents_top - Maximum Number of Associated Agents" on page 218
	"authority_lvl - User Authorization Level" on page 200 (deprecated)
	"authority_bitmap - User Authorization Level monitor element" on page 201
	"binds_precompiles - Binds/Precompiles Attempted" on page 376
	"cat_cache_inserts - Catalog Cache Inserts" on page 278
	"cat_cache_lookups - Catalog Cache Lookups" on page 277
	"cat_cache_overflows - Catalog Cache Overflows" on page 278
	"commit_sql_stmts - Commit Statements Attempted" on page 369
	"conn_complete_time - Connection Request Completion Timestamp" on page 204
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 372
	"deadlocks - Deadlocks Detected" on page 303
	"direct_read_reqs - Direct Read Requests" on page 274
	"direct_read_time - Direct Read Time" on page 275
	"direct_reads - Direct Reads From Database" on page 273
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_write_time - Direct Write Time" on page 276
	"direct_writes - Direct Writes to Database" on page 273
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 368
	"failed_sql_stmts - Failed Statement Operations" on page 368
	"hash_join_overflows - Hash Join Overflows" on page 233
	"hash_join_small_overflows - Hash Join Small Overflows" on page 234
	"inbound_comm_address - Inbound Communication Address" on page 441
	"int_auto_rebinds - Internal Automatic Rebinds" on page 373
	"int_commits - Internal Commits" on page 373
	"int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock" on page 375
	"int_rollbacks - Internal Rollbacks" on page 374
	"int_rows_deleted - Internal Rows Deleted" on page 353
	"int_rows_inserted - Internal Rows Inserted" on page 355
	"int_rows_updated - Internal Rows Updated" on page 354
	"last_reset - Last Reset Timestamp" on page 409
	"lock_escalation - Lock Escalation" on page 310
	"lock_timeouts - Number of Lock Timeouts" on page 309
	"lock_timeout_val - Lock timeout" on page 320
	"lock_wait_time - Time Waited On Locks" on page 318
	"lock_waits - Lock Waits" on page 317

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 301</p> <p>"locks_waiting - Current Agents Waiting On Locks" on page 318</p> <p>"num_agents - Number of Agents Working on a Statement" on page 403</p> <p>"olap_func_overflows - OLAP Function Overflows monitor element" on page 235</p> <p>"open_loc_curs - Open Local Cursors" on page 366</p> <p>"open_loc_curs_blk - Open Local Cursors with Blocking" on page 366</p> <p>"open_rem_curs - Open Remote Cursors" on page 364</p> <p>"open_rem_curs_blk - Open Remote Cursors with Blocking" on page 365</p> <p>"pkg_cache_inserts - Package Cache Inserts" on page 281</p> <p>"pkg_cache_lookups - Package Cache Lookups" on page 280</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243</p> <p>"pool_data_writes - Buffer Pool Data Writes" on page 245</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248</p> <p>"pool_index_writes - Buffer Pool Index Writes" on page 249</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time" on page 255</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244</p> <p>"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247</p> <p>"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249</p> <p>"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 252</p> <p>"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 253</p> <p>"pool_write_time - Total Buffer Pool Physical Write Time" on page 255</p> <p>"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 250</p> <p>"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 253</p> <p>"pool_xda_writes - Buffer Pool XDA Data Writes" on page 254</p> <p>"prefetch_wait_time - Time Waited for Prefetch" on page 268</p> <p>"prev_uow_stop_time - Previous Unit of Work Completion Timestamp" on page 205</p> <p>"priv_workspace_num_overflows - Private Workspace Overflows" on page 287</p> <p>"priv_workspace_section_inserts - Private Workspace Section Inserts" on page 288</p> <p>"priv_workspace_section_lookups - Private Workspace Section Lookups" on page 287</p> <p>"priv_workspace_size_top - Maximum Private Workspace Size" on page 286</p> <p>"rej_curs_blk - Rejected Block Cursor Requests" on page 365</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 370</p> <p>"rows_deleted - Rows Deleted" on page 349</p> <p>"rows_inserted - Rows Inserted" on page 349</p> <p>"rows_read - Rows Read" on page 352</p> <p>"rows_selected - Rows Selected" on page 350</p> <p>"rows_updated - Rows Updated" on page 350</p> <p>"rows_written - Rows Written" on page 351</p> <p>"select_sql_stmts - Select SQL Statements Executed" on page 371</p> <p>"shr_workspace_num_overflows - Shared Workspace Overflows" on page 284</p> <p>"shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 285</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 285
	"shr_workspace_size_top - Maximum Shared Workspace Size" on page 283
	"sort_overflows - Sort Overflows" on page 228
	"sql_reqs_since_commit - SQL Requests Since Last Commit" on page 376
	"static_sql_stmts - Static SQL Statements Attempted" on page 367
	"total_hash_joins - Total Hash Joins" on page 231
	"total_hash_loops - Total Hash Loops" on page 233
	"total_olap_funcs - Total OLAP Functions monitor element" on page 235
	"total_sort_time - Total Sort Time" on page 227
	"total_sorts - Total Sorts" on page 227
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 371
	"unread_prefetch_pages - Unread Prefetch Pages" on page 269
	"uow_comp_status - Unit of Work Completion Status" on page 207
	"uow_elapsed_time - Most Recent Unit of Work Elapsed Time" on page 206
	"uow_lock_wait_time - Total Time Unit of Work Waited on Locks" on page 319
	"uow_log_space_used - Unit of Work Log Space Used" on page 293
	"uow_start_time - Unit of Work Start Timestamp" on page 205
	"uow_stop_time - Unit of Work Stop Timestamp" on page 206
	"x_lock_escalations - Exclusive Lock Escalations" on page 305
	"xquery_stmts - XQuery Statements Attempted" on page 377
appl_id_info	"agent_id - Application Handle (agent ID)" on page 184
	"appl_id - Application ID" on page 189
	"appl_name - Application Name" on page 189
	"appl_status - Application Status" on page 185
	"auth_id - Authorization ID" on page 192
	"client_db_alias - Database Alias Used by Application" on page 194
	"client_prdid - Client Product/Version ID" on page 193
	"codepage_id - ID of Code Page Used by Application" on page 187
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
	"input_db_alias - Input Database Alias" on page 409
	"sequence_no - Sequence number monitor element" on page 191
	"status_change_time - Application Status Change Time" on page 187

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 184
	"appl_id - Application ID" on page 189
	"appl_name - Application Name" on page 189
	"appl_section_inserts - Section Inserts monitor element" on page 289
	"appl_section_lookups - Section Lookups" on page 289
	"appl_status - Application Status" on page 185
	"auth_id - Authorization ID" on page 192
	"authority_lvl - User Authorization Level" on page 200 (deprecated)
	"authority_bitmap - User Authorization Level monitor element" on page 201
	"client_db_alias - Database Alias Used by Application" on page 194
	"client_pid - Client Process ID" on page 197
	"client_platform - Client Operating Platform" on page 198
	"client_prdid - Client Product/Version ID" on page 193
	"client_protocol - Client Communication Protocol" on page 198
	"codepage_id - ID of Code Page Used by Application" on page 187
	"coord_agent_pid - Coordinator Agent monitor element" on page 209
	"coord_node - Coordinating Node" on page 203
	"corr_token - DRDA Correlation Token" on page 197
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
	"execution_id - User Login ID" on page 196
	"input_db_alias - Input Database Alias" on page 409
	"is_system_appl - Is System Application monitor element" on page 195
	"num_assoc_agents - Number of Associated Agents" on page 219
	"sequence_no - Sequence number monitor element" on page 191
	"status_change_time - Application Status Change Time" on page 187
	"territory_code - Database Territory Code" on page 199
	"tpmon_acc_str - TP Monitor Client Accounting String" on page 464
	"tpmon_client_app - TP Monitor Client Application Name" on page 464
	"tpmon_client_userid - TP Monitor Client User ID" on page 463
	"tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 463
	"workload_id - Workload ID monitor element" on page 504
	"session_auth_id - Session Authorization ID" on page 193

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 184
	"appl_id - Application ID" on page 189
	"appl_name - Application Name" on page 189
	"appl_status - Application Status" on page 185
	"auth_id - Authorization ID" on page 192
	"client_db_alias - Database Alias Used by Application" on page 194
	"codepage_id - ID of Code Page Used by Application" on page 187
	"locks_held - Locks Held" on page 301
	"locks_waiting - Current Agents Waiting On Locks" on page 318
	"lock_wait_time - Time Waited On Locks" on page 318
	"sequence_no - Sequence number monitor element" on page 191
	"session_auth_id - Session Authorization ID" on page 193
	"status_change_time - Application Status Change Time" on page 187
	appl_remote
"create_nickname - Create Nicknames" on page 467	
"create_nickname_time - Create Nickname Response Time" on page 472	
"datasource_name - Data Source Name" on page 465	
"db_name - Database Name" on page 176	
"delete_sql_stmts - Deletes" on page 467	
"delete_time - Delete Response Time" on page 471	
"failed_sql_stmts - Failed Statement Operations" on page 368	
"insert_sql_stmts - Inserts" on page 465	
"insert_time - Insert Response Time" on page 470	
"passthru_time - Pass-Through Time" on page 472	
"passthru - Pass-Through" on page 468	
"remote_lock_time - Remote Lock Time" on page 473	
"remote_locks - Remote Locks" on page 468	
"rollback_sql_stmts - Rollback Statements Attempted" on page 370	
"rows_deleted - Rows Deleted" on page 349	
"rows_inserted - Rows Inserted" on page 349	
"rows_selected - Rows Selected" on page 350	
"rows_updated - Rows Updated" on page 350	
"select_sql_stmts - Select SQL Statements Executed" on page 371	
"select_time - Query Response Time" on page 469	
"sp_rows_selected - Rows Returned by Stored Procedures" on page 469	
"stored_proc_time - Stored Procedure Time" on page 473	
"stored_procs - Stored Procedures" on page 468	
"update_sql_stmts - Updates" on page 466	
"update_time - Update Response Time" on page 471	

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 270
	"bp_name - Buffer Pool Name" on page 268
	"bp_id - Buffer pool identifier monitor element" on page 241
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
	"direct_read_reqs - Direct Read Requests" on page 274
	"direct_reads - Direct Reads From Database" on page 273
	"direct_read_time - Direct Read Time" on page 275
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_writes - Direct Writes to Database" on page 273
	"direct_write_time - Direct Write Time" on page 276
	"files_closed - Database Files Closed" on page 256
	"input_db_alias - Input Database Alias" on page 409
	"pages_from_block_ios - Total Number of Pages Read by Block IO" on page 271
	"pages_from_vectored_ios - Total Number of Pages Read by Vectored IO" on page 270
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 263
	"pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 257
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 258
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 263
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 259
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 258
	"pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 261
	"pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 262
	"pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests" on page 264
	"pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads" on page 260
	"pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes" on page 261
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_data_writes - Buffer Pool Data Writes" on page 245
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_index_writes - Buffer Pool Index Writes" on page 249
	"pool_read_time - Total Buffer Pool Physical Read Time" on page 255
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 266
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 252
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 253

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

Snapshot logical data groups	Monitor element
bufferpool (continued)	<p>"pool_write_time - Total Buffer Pool Physical Write Time" on page 255</p> <p>"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 250</p> <p>"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 253</p> <p>"pool_xda_writes - Buffer Pool XDA Data Writes" on page 254</p> <p>"vectored_ios - Number of Vectored IO Requests" on page 269</p>
bufferpool_nodeinfo	<p>"bp_cur_buffsz - Current Size of Buffer Pool" on page 271</p> <p>"bp_new_buffsz - New Buffer Pool Size" on page 272</p> <p>"bp_pages_left_to_remove - Number of Pages Left to Remove" on page 272</p> <p>"bp_tbsp_use_count - Number of Table Spaces Mapped to Buffer Pool" on page 272</p> <p>"node_number - Node Number" on page 202</p>
collected	<p>"node_number - Node Number" on page 202</p> <p>"server_db2_type - Database Manager Type at Monitored (Server) Node" on page 173</p> <p>"server_instance_name - Server Instance Name" on page 172</p> <p>"server_prdid - Server Product/Version ID" on page 173</p> <p>"server_version - Server Version" on page 174</p> <p>switch_list Monitor switches control data</p> <p>"time_stamp - Snapshot Time" on page 410</p> <p>"time_zone_disp - Time Zone Displacement" on page 176</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 217	
	"agents_from_pool - Agents Assigned From Pool" on page 216	
	"agents_registered - Agents Registered" on page 214	
	"agents_registered_top - Maximum Number of Agents Registered" on page 215	
	"agents_stolen - Stolen Agents" on page 218	
	"agents_waiting_on_token - Agents Waiting for a Token" on page 214	
	"agents_waiting_top - Maximum Number of Agents Waiting monitor element" on page 215	
	"comm_private_mem - Committed Private Memory" on page 218	
	"con_local_databases - Local Databases with Current Connects" on page 212	
	"coord_agents_top - Maximum Number of Coordinating Agents" on page 217	
	"db2start_time - Start Database Manager Timestamp" on page 172	
	"db_status - Status of Database" on page 179	
	"gw_total_cons - Total Number of Attempted Connections for DB2 Connect" on page 435	
	"gw_cur_cons - Current Number of Connections for DB2 Connect" on page 436	
	"gw_cons_wait_host - Number of Connections Waiting for the Host to Reply" on page 436	
	"gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request" on page 437	
	"idle_agents - Number of Idle Agents" on page 216	
	"last_reset - Last Reset Timestamp" on page 409	
	"local_cons - Local Connections" on page 211	
	"local_cons_in_exec - Local Connections Executing in the Database Manager" on page 212	
	"max_agent_overflows - Maximum Agent Overflows" on page 219	
	"num_gw_conn_switches - Connection Switches" on page 220	
	"num_nodes_in_db2_instance - Number of Nodes in Partition" on page 410	
	"piped_sorts_requested - Piped Sorts Requested" on page 225	
	"piped_sorts_accepted - Piped Sorts Accepted" on page 226	
	"post_threshold_hash_joins - Hash Join Threshold" on page 232	
	"post_threshold_olap_funcs - OLAP Function Threshold monitor element" on page 236	
	"post_threshold_sorts - Post Threshold Sorts" on page 224	
	"product_name - Product Name" on page 175	
	"rem_cons_in - Remote Connections To Database Manager" on page 210	
	"rem_cons_in_exec - Remote Connections Executing in the Database Manager" on page 211	
	"service_level - Service Level" on page 174	
	"smallest_log_avail_node - Node with Least Available Log Space" on page 188	
	"sort_heap_allocated - Total Sort Heap Allocated" on page 224	
	"sort_heap_top - Sort private heap high watermark" on page 230	
	switch_list Monitor switches control data	
	db_lock_list	"appls_cur_cons - Applications Connected Currently" on page 213
		"db_name - Database Name" on page 176
		"db_path - Database Path" on page 177
		"input_db_alias - Input Database Alias" on page 409
"locks_held - Locks Held" on page 301		
"locks_waiting - Current Agents Waiting On Locks" on page 318		

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 233
	"active_olap_funcs - Active OLAP Functions monitor element" on page 236
	"active_sorts - Active Sorts" on page 229
	"agents_top - Number of Agents Created" on page 403
	"appl_id_oldest_xact - Application with Oldest Transaction" on page 188
	"appl_section_inserts - Section Inserts monitor element" on page 289
	"appl_section_lookups - Section Lookups" on page 289
	"appls_cur_cons - Applications Connected Currently" on page 213
	"appls_in_db2 - Applications Executing in the Database Currently" on page 214
	"async_runstats - Total number of asynchronous RUNSTATS requests monitor element" on page 507
	"binds_precompiles - Binds/Precompiles Attempted" on page 376
	"blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element" on page 301
	"cat_cache_inserts - Catalog Cache Inserts" on page 278
	"cat_cache_lookups - Catalog Cache Lookups" on page 277
	"cat_cache_overflows - Catalog Cache Overflows" on page 278
	"cat_cache_size_top - Catalog cache high watermark" on page 279
	"catalog_node - Catalog Node Number" on page 180
	"catalog_node_name - Catalog Node Network Name" on page 179
	"commit_sql_stmts - Commit Statements Attempted" on page 369
	"connections_top - Maximum Number of Concurrent Connections" on page 204
	"coord_agents_top - Maximum Number of Coordinating Agents" on page 217
	"db_conn_time - Database Activation Timestamp" on page 178
	"db_heap_top - Maximum Database Heap Allocated" on page 290
	"db_location - Database Location" on page 180
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
	"db_status - Status of Database" on page 179
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 372
	"deadlocks - Deadlocks Detected" on page 303
	"direct_read_reqs - Direct Read Requests" on page 274
	"direct_read_time - Direct Read Time" on page 275
	"direct_reads - Direct Reads From Database" on page 273
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_write_time - Direct Write Time" on page 276
	"direct_writes - Direct Writes to Database" on page 273
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 368
	"failed_sql_stmts - Failed Statement Operations" on page 368

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 256
	"hash_join_overflows - Hash Join Overflows" on page 233
	"hash_join_small_overflows - Hash Join Small Overflows" on page 234
	"input_db_alias - Input Database Alias" on page 409
	"int_auto_rebinds - Internal Automatic Rebinds" on page 373
	"int_commits - Internal Commits" on page 373
	"int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock" on page 375
	"int_rollbacks - Internal Rollbacks" on page 374
	"int_rows_deleted - Internal Rows Deleted" on page 353
	"int_rows_inserted - Internal Rows Inserted" on page 355
	"int_rows_updated - Internal Rows Updated" on page 354
	"last_backup - Last Backup Timestamp" on page 180
	"last_reset - Last Reset Timestamp" on page 409
	"lock_escals - Number of Lock Escalations" on page 304
	"lock_list_in_use - Total Lock List Memory In Use" on page 302
	"lock_timeouts - Number of Lock Timeouts" on page 309
	"lock_wait_time - Time Waited On Locks" on page 318
	"lock_waits - Lock Waits" on page 317
	"locks_held - Locks Held" on page 301
	"locks_waiting - Current Agents Waiting On Locks" on page 318
	"log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages" on page 295
	"log_read_time - Log Read Time" on page 296
	"log_reads - Number of Log Pages Read" on page 292
	"log_to_redo_for_recovery - Amount of Log to be Redone for Recovery" on page 296
	"log_write_time - Log Write Time" on page 296
	"log_writes - Number of Log Pages Written" on page 293
	"num_assoc_agents - Number of Associated Agents" on page 219
	"num_db_storage_paths - Number of automatic storage paths" on page 181
	"num_indoubt_trans - Number of Indoubt Transactions" on page 316
	"num_log_buffer_full - Number of Full Log Buffers" on page 298
	"num_log_data_found_in_buffer - Number of Log Data Found In Buffer" on page 299
	"num_log_part_page_io - Number of Partial Log Page Writes" on page 298
	"num_log_read_io - Number of Log Reads" on page 297
	"num_log_write_io - Number of Log Writes" on page 297
	"olap_func_overflows - OLAP Function Overflows monitor element" on page 235

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 281
	"pkg_cache_lookups - Package Cache Lookups" on page 280
	"pkg_cache_num_overflows - Package Cache Overflows" on page 282
	"pkg_cache_size_top - Package cache high watermark" on page 282
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 263
	"pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 257
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 258
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 263
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 259
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 258
	"pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 261
	"pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests" on page 264
	"pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads" on page 260
	"pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes" on page 261
	"pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 262
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_data_writes - Buffer Pool Data Writes" on page 245
	"pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered" on page 265
	"pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered" on page 267
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_index_writes - Buffer Pool Index Writes" on page 249
	"pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered" on page 265
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 266
	"pool_read_time - Total Buffer Pool Physical Read Time" on page 255
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 252
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 253
	"pool_write_time - Total Buffer Pool Physical Write Time" on page 255
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 250
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 253
	"pool_xda_writes - Buffer Pool XDA Data Writes" on page 254
	"post_shrthreshold_hash_joins - Post threshold hash joins" on page 232
	"post_shrthreshold_sorts - Post shared threshold sorts" on page 225
	"priv_workspace_num_overflows - Private Workspace Overflows" on page 287
	"priv_workspace_section_inserts - Private Workspace Section Inserts" on page 288
	"priv_workspace_section_lookups - Private Workspace Section Lookups" on page 287

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 286
	"rollback_sql_stmts - Rollback Statements Attempted" on page 370
	"rows_deleted - Rows Deleted" on page 349
	"rows_inserted - Rows Inserted" on page 349
	"rows_read - Rows Read" on page 352
	"rows_selected - Rows Selected" on page 350
	"rows_updated - Rows Updated" on page 350
	"sec_log_used_top - Maximum Secondary Log Space Used" on page 290
	"sec_logs_allocated - Secondary Logs Allocated Currently" on page 292
	"select_sql_stmts - Select SQL Statements Executed" on page 371
	"server_platform - Server Operating System" on page 175
	"shr_workspace_num_overflows - Shared Workspace Overflows" on page 284
	"shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 285
	"shr_workspace_section_lookups - Shared Workspace Section Lookups" on page 285
	"shr_workspace_size_top - Maximum Shared Workspace Size" on page 283
	"sort_heap_allocated - Total Sort Heap Allocated" on page 224
	"sort_overflows - Sort Overflows" on page 228
	"sort_shrheap_allocated - Sort Share Heap Currently Allocated" on page 230
	"sort_shrheap_top - Sort share heap high watermark" on page 230
	"static_sql_stmts - Static SQL Statements Attempted" on page 367
	"stats_cache_size - Size of statistics cache monitor element" on page 506
	"stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 508
	"stats_fabrications - Total number of statistics fabrications monitor elements" on page 506
	"sync_runstats - Total number of synchronous RUNSTATS activities monitor element" on page 507
	"sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 509
	"tot_log_used_top - Maximum Total Log Space Used" on page 291
	"total_cons - Connects Since Database Activation" on page 213
	"total_hash_joins - Total Hash Joins" on page 231
	"total_hash_loops - Total Hash Loops" on page 233
	"total_log_available - Total Log Available" on page 294
	"total_log_used - Total Log Space Used" on page 294
	"total_olap_funcs - Total OLAP Functions monitor element" on page 235
	"total_sec_cons - Secondary Connections" on page 219
	"total_sort_time - Total Sort Time" on page 227
	"total_sorts - Total Sorts" on page 227
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 371
	"unread_prefetch_pages - Unread Prefetch Pages" on page 269
	"x_lock_escalations - Exclusive Lock Escalations" on page 305
	"xquery_stmts - XQuery Statements Attempted" on page 377

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 369	
	"create_nickname - Create Nicknames" on page 467	
	"create_nickname_time - Create Nickname Response Time" on page 472	
	"datasource_name - Data Source Name" on page 465	
	"db_name - Database Name" on page 176	
	"delete_sql_stmts - Deletes" on page 467	
	"delete_time - Delete Response Time" on page 471	
	"disconnects - Disconnects" on page 465	
	"failed_sql_stmts - Failed Statement Operations" on page 368	
	"insert_sql_stmts - Inserts" on page 465	
	"insert_time - Insert Response Time" on page 470	
	"passthru_time - Pass-Through Time" on page 472	
	"passthru - Pass-Through" on page 468	
	"remote_lock_time - Remote Lock Time" on page 473	
	"remote_locks - Remote Locks" on page 468	
	"rollback_sql_stmts - Rollback Statements Attempted" on page 370	
	"rows_deleted - Rows Deleted" on page 349	
	"rows_inserted - Rows Inserted" on page 349	
	"rows_selected - Rows Selected" on page 350	
	"rows_updated - Rows Updated" on page 350	
	"select_sql_stmts - Select SQL Statements Executed" on page 371	
	"select_time - Query Response Time" on page 469	
	"sp_rows_selected - Rows Returned by Stored Procedures" on page 469	
	"stored_proc_time - Stored Procedure Time" on page 473	
	"stored_procs - Stored Procedures" on page 468	
	"total_cons - Connects Since Database Activation" on page 213	
	"update_sql_stmts - Updates" on page 466	
	"update_time - Update Response Time" on page 471	
	db_storage_group	"db_storage_path - Automatic storage path" on page 181
		"sto_path_free_sz - Automatic Storage Path Free Space" on page 181
"fs_used_size - Amount of Space Used on a File System" on page 182		
"fs_total_size - Total Size of a File System" on page 182		
"fs_id - Unique File System Identification Number" on page 183		
"fs_type - File System Type" on page 183		
"node_number - Node Number" on page 202		

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

Snapshot logical data groups	Monitor element
dcs_appl	"appl_idle_time - Application Idle Time" on page 208
	"commit_sql_stmts - Commit Statements Attempted" on page 369
	"elapsed_exec_time - Statement Execution Elapsed Time" on page 458
	"failed_sql_stmts - Failed Statement Operations" on page 368
	"gw_con_time - DB2 Connect Gateway First Connect Initiated" on page 435
	"gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing" on page 437
	"host_response_time - Host Response Time" on page 459
	"inbound_bytes_received - Inbound Number of Bytes Received" on page 442
	"inbound_bytes_sent - Inbound Number of Bytes Sent" on page 443
	"last_reset - Last Reset Timestamp" on page 409
	"max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 445
	"max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 446
	"max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 447
	"max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 448
	"max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 449
	"max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 450
	"max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 450
	"max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 451
	"max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element" on page 452
	"max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element" on page 453
"max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 454	

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

Snapshot logical data groups	Monitor element
dc_s_appl (continued)	<p>"max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 445</p> <p>"max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 446</p> <p>"max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 447</p> <p>"max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 447</p> <p>"max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 448</p> <p>"max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 449</p> <p>"max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 450</p> <p>"max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 451</p> <p>"max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 452</p> <p>"max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 453</p> <p>"max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 453</p> <p>"max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms" on page 454</p> <p>"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms" on page 455</p> <p>"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms" on page 455</p> <p>"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms" on page 456</p> <p>"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms" on page 456</p> <p>"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms" on page 457</p> <p>"network_time_top - Maximum Network Time for Statement" on page 457</p> <p>"network_time_bottom - Minimum Network Time for Statement" on page 457</p> <p>"open_cursors - Number of Open Cursors" on page 439</p> <p>"outbound_bytes_received - Outbound Number of Bytes Received" on page 442</p> <p>"outbound_bytes_sent - Outbound Number of Bytes Sent" on page 442</p> <p>"prev_uow_stop_time - Previous Unit of Work Completion Timestamp" on page 205</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 370</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 350
	"sql_stmts - Number of SQL Statements Attempted" on page 437
	"tpmon_acc_str - TP Monitor Client Accounting String" on page 464
	"tpmon_client_app - TP Monitor Client Application Name" on page 464
	"tpmon_client_userid - TP Monitor Client User ID" on page 463
	"tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 463
	"uow_comp_status - Unit of Work Completion Status" on page 207
	"uow_elapsed_time - Most Recent Unit of Work Elapsed Time" on page 206
	"uow_start_time - Unit of Work Start Timestamp" on page 205
	"uow_stop_time - Unit of Work Stop Timestamp" on page 206
	"xid - Transaction ID" on page 458
dcs_appl_info	"agent_id - Application Handle (agent ID)" on page 184
	"agent_status - DCS Application Agents" on page 440
	"appl_id - Application ID" on page 189
	"appl_name - Application Name" on page 189
	"auth_id - Authorization ID" on page 192
	"client_pid - Client Process ID" on page 197
	"client_platform - Client Operating Platform" on page 198
	"client_prdid - Client Product/Version ID" on page 193
	"client_protocol - Client Communication Protocol" on page 198
	"codepage_id - ID of Code Page Used by Application" on page 187
	"dcs_appl_status - DCS Application Status" on page 439
	"dcs_db_name - DCS Database Name" on page 434
	"execution_id - User Login ID" on page 196
	"gw_db_alias - Database Alias at the Gateway" on page 434
	"host_ccsid - Host Coded Character Set ID" on page 440
	"host_db_name - Host Database Name" on page 434
	"host_prdid - Host Product/Version ID" on page 194
	"inbound_comm_address - Inbound Communication Address" on page 441
	"outbound_appl_id - Outbound Application ID" on page 195
	"outbound_comm_address - Outbound Communication Address" on page 441
	"outbound_comm_protocol - Outbound Communication Protocol" on page 441
	"outbound_sequence_no - Outbound Sequence Number" on page 196
	"sequence_no - Sequence number monitor element" on page 191
"status_change_time - Application Status Change Time" on page 187	

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 369</p> <p>"con_elapsed_time - Most Recent Connection Elapsed Time" on page 461</p> <p>"con_response_time - Most Recent Response Time for Connect" on page 461</p> <p>"dcs_db_name - DCS Database Name" on page 434</p> <p>"elapsed_exec_time - Statement Execution Elapsed Time" on page 458</p> <p>"failed_sql_stmts - Failed Statement Operations" on page 368</p> <p>"gw_comm_error_time - Communication Error Time" on page 462</p> <p>"gw_comm_errors - Communication Errors" on page 461</p> <p>"gw_con_time - DB2 Connect Gateway First Connect Initiated" on page 435</p> <p>"gw_connections_top - Maximum Number of Concurrent Connections to Host Database" on page 435</p> <p>"gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request" on page 437</p> <p>"gw_cons_wait_host - Number of Connections Waiting for the Host to Reply" on page 436</p> <p>"gw_cur_cons - Current Number of Connections for DB2 Connect" on page 436</p> <p>"gw_total_cons - Total Number of Attempted Connections for DB2 Connect" on page 435</p> <p>"host_db_name - Host Database Name" on page 434</p> <p>"host_response_time - Host Response Time" on page 459</p> <p>"inbound_bytes_received - Inbound Number of Bytes Received" on page 442</p> <p>"last_reset - Last Reset Timestamp" on page 409</p> <p>"max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 445</p> <p>"max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 446</p> <p>"max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 447</p> <p>"max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 448</p> <p>"max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 449</p> <p>"max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 450</p> <p>"max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 450</p> <p>"max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 451</p> <p>"max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element" on page 452</p> <p>"max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element" on page 453</p> <p>"max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 454</p>

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

Snapshot logical data groups	Monitor element
dc_s_dbase (continued)	<p>"max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 445</p> <p>"max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 446</p> <p>"max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 447</p> <p>"max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 447</p> <p>"max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 448</p> <p>"max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 449</p> <p>"max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 450</p> <p>"max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 451</p> <p>"max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 452</p> <p>"max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 453</p> <p>"max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 453</p> <p>"max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms" on page 454</p> <p>"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms" on page 455</p> <p>"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms" on page 455</p> <p>"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms" on page 456</p> <p>"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms" on page 456</p> <p>"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms" on page 457</p> <p>"network_time_bottom - Minimum Network Time for Statement" on page 457</p> <p>"network_time_top - Maximum Network Time for Statement" on page 457</p> <p>"outbound_bytes_sent - Outbound Number of Bytes Sent" on page 442</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 370</p> <p>"rows_selected - Rows Selected" on page 350</p> <p>"sql_stmts - Number of SQL Statements Attempted" on page 437</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 462
	"creator - Application Creator" on page 382
	"elapsed_exec_time - Statement Execution Elapsed Time" on page 458
	"fetch_count - Number of Successful Fetches" on page 386
	"gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing" on page 437
	"host_response_time - Host Response Time" on page 459
	"inbound_bytes_received - Inbound Number of Bytes Received" on page 442
	"inbound_bytes_sent - Inbound Number of Bytes Sent" on page 443
	"num_transmissions_group - Number of Transmissions Group" on page 460
	"num_transmissions - Number of Transmissions" on page 460
	"outbound_bytes_received - Outbound Number of Bytes Received" on page 442
	"outbound_bytes_sent - Outbound Number of Bytes Sent" on page 442
	"package_name - Package Name" on page 379
	"query_card_estimate - Query Number of Rows Estimate" on page 387
	"query_cost_estimate - Query Cost Estimate" on page 388
	"section_number - Section Number" on page 381
	"stmt_elapsed_time - Most Recent Statement Elapsed Time" on page 384
	"stmt_operation/operation - Statement Operation" on page 378
	"stmt_start - Statement Operation Start Timestamp" on page 382
	"stmt_stop - Statement Operation Stop Timestamp" on page 383
"stmt_text - SQL Statement Text monitor element" on page 384	
detail_log	"current_active_log - Current Active Log File Number" on page 300
	"current_archive_log - Current Archive Log File Number" on page 300
	"first_active_log - First Active Log File Number" on page 299
	"last_active_log - Last Active Log File Number" on page 299
	"node_number - Node Number" on page 202

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 386
	"int_rows_deleted - Internal Rows Deleted" on page 353
	"int_rows_inserted - Internal Rows Inserted" on page 355
	"int_rows_updated - Internal Rows Updated" on page 354
	"num_compilations - Statement Compilations" on page 401
	"num_executions - Statement Executions" on page 401
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 252
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 253
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 250
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 253
	"prep_time_best - Statement best preparation time monitor element" on page 402
	"prep_time_worst - Statement worst preparation time monitor element" on page 402
	"rows_read - Rows Read" on page 352
	"rows_written - Rows Written" on page 351
	"sort_overflows - Sort Overflows" on page 228
	"stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 508
	"stmt_pkgcache_id - Statement package cache identifier" on page 393
	"stmt_sorts - Statement Sorts" on page 385
	"stmt_text - SQL Statement Text monitor element" on page 384
	"sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 509
	"total_exec_time - Elapsed Statement Execution Time" on page 402
	"total_sort_time - Total Sort Time" on page 227
	"total_sys_cpu_time - Total System CPU for a Statement" on page 408
	"total_usr_cpu_time - Total User CPU for a Statement" on page 408
	"insert_timestamp - Statement insert timestamp monitor element" on page 384
dynsql_list	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
fcm	"buff_free - FCM Buffers Currently Free" on page 237
	"buff_free_bottom - Minimum FCM Buffers Free" on page 237
	"ch_free - Channels Currently Free" on page 239
	"ch_free_bottom - Minimum Channels Free" on page 239

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 238
	"node_number - Node Number" on page 202
	"total_buffers_sent - Total FCM Buffers Sent" on page 238
	"total_buffers_rcvd - Total FCM Buffers Received" on page 238
hadr	"hadr_connect_status - HADR Connection Status monitor element" on page 424
	"hadr_connect_time - HADR Connection Time monitor element" on page 425
	"hadr_heartbeat - HADR Heartbeat monitor element" on page 426
	"hadr_local_host - HADR Local Host monitor element" on page 427
	"hadr_local_service - HADR Local Service monitor element" on page 427
	"hadr_log_gap - HADR Log Gap" on page 432
	"hadr_primary_log_file - HADR Primary Log File monitor element" on page 429
	"hadr_primary_log_lsn - HADR Primary Log LSN monitor element" on page 430
	"hadr_primary_log_page - HADR Primary Log Page monitor element" on page 430
	"hadr_remote_host - HADR Remote Host monitor element" on page 428
	"hadr_remote_instance - HADR Remote Instance monitor element" on page 429
	"hadr_remote_service - HADR Remote Service monitor element" on page 428
	"hadr_role - HADR Role" on page 422
	"hadr_standby_log_file - HADR Standby Log File monitor element" on page 431
	"hadr_standby_log_lsn - HADR Standby Log LSN monitor element" on page 432
	"hadr_standby_log_page - HADR Standby Log Page monitor element" on page 431
"hadr_state - HADR State monitor element" on page 423	
"hadr_syncmode - HADR Synchronization Mode monitor element" on page 424	
"hadr_timeout - HADR Timeout monitor element" on page 429	
lock	"data_partition_id - Data partition identifier monitor element" on page 302
	"lock_attributes - Lock Attributes" on page 314
	"lock_count - Lock Count" on page 315
	"lock_current_mode - Original Lock Mode Before Conversion" on page 316
	"lock_escalation - Lock Escalation" on page 310
	"lock_hold_count - Lock Hold Count" on page 316
	"lock_mode - Lock Mode" on page 306
	"lock_name - Lock Name" on page 313
	"lock_object_name - Lock Object Name" on page 308
	"lock_object_type - Lock Object Type Waited On" on page 307
	"lock_release_flags - Lock Release Flags" on page 315
	"lock_status - Lock Status" on page 307
	"node_number - Node Number" on page 202
	"table_file_id - Table File ID" on page 355
	"table_name - Table Name" on page 347
"table_schema - Table Schema Name" on page 348	
"tablespace_name - Table Space Name" on page 325	

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

Snapshot logical data groups	Monitor element
lock_wait	<p>"agent_id_holding_lock - Agent ID Holding Lock" on page 320</p> <p>"appl_id_holding_lk - Application ID Holding Lock" on page 321</p> <p>"lock_attributes - Lock Attributes" on page 314</p> <p>"lock_current_mode - Original Lock Mode Before Conversion" on page 316</p> <p>"lock_escalation - Lock Escalation" on page 310</p> <p>"lock_mode - Lock Mode" on page 306</p> <p>"lock_mode_requested - Lock Mode Requested" on page 311</p> <p>"lock_name - Lock Name" on page 313</p> <p>"lock_object_type - Lock Object Type Waited On" on page 307</p> <p>"lock_release_flags - Lock Release Flags" on page 315</p> <p>"lock_wait_start_time - Lock Wait Start Timestamp" on page 319</p> <p>"node_number - Node Number" on page 202</p> <p>"ss_number - Subsection Number" on page 396</p> <p>"table_name - Table Name" on page 347</p> <p>"table_schema - Table Schema Name" on page 348</p> <p>"tablespace_name - Table Space Name" on page 325</p> <p>"data_partition_id - Data partition identifier monitor element" on page 302</p>
memory_pool	<p>"node_number - Node Number" on page 202</p> <p>"pool_cur_size - Current Size of Memory Pool" on page 222</p> <p>"pool_id - Memory Pool Identifier" on page 220</p> <p>"pool_secondary_id - Memory Pool Secondary Identifier" on page 221</p> <p>"pool_config_size - Configured Size of Memory Pool" on page 222</p> <p>"pool_watermark - Memory Pool Watermark" on page 223</p>
progress	<p>"progress_completed_units - Completed Progress Work Units" on page 421</p> <p>"progress_description - Progress Description" on page 419</p> <p>"progress_seq_num - Progress Sequence Number" on page 419</p> <p>"progress_start_time - Progress Start Time" on page 420</p> <p>"progress_total_units - Total Progress Work Units" on page 420</p> <p>"progress_work_metric - Progress Work Metric" on page 420</p>
progress_list	<p>"progress_list_cur_seq_num - Current Progress List Sequence Number" on page 419</p> <p>"progress_list_attr - Current Progress List Attributes" on page 421</p>
rollforward	<p>"node_number - Node Number" on page 202</p> <p>"rf_type - Rollforward Type" on page 323</p> <p>"rf_log_num - Log Being Rolled Forward" on page 324</p> <p>"rf_status - Log Phase" on page 324</p> <p>"rf_timestamp - Rollforward Timestamp" on page 323</p> <p>"ts_name - Tablespace Being Rolled Forward" on page 323</p>

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 403
	"blocking_cursor - Blocking Cursor" on page 462
	"consistency_token - Package Consistency Token" on page 380
	"creator - Application Creator" on page 382
	"cursor_name - Cursor Name" on page 381
	"degree_parallelism - Degree of Parallelism" on page 403
	"fetch_count - Number of Successful Fetches" on page 386
	"int_rows_deleted - Internal Rows Deleted" on page 353
	"int_rows_inserted - Internal Rows Inserted" on page 355
	"int_rows_updated - Internal Rows Updated" on page 354
	"num_agents - Number of Agents Working on a Statement" on page 403
	"package_name - Package Name" on page 379
	"package_version_id - Package Version" on page 380
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 252
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 253
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 250
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 253
	"query_card_estimate - Query Number of Rows Estimate" on page 387
	"query_cost_estimate - Query Cost Estimate" on page 388
	"rows_read - Rows Read" on page 352
	"rows_written - Rows Written" on page 351
	"section_number - Section Number" on page 381
	"sort_overflows - Sort Overflows" on page 228
	"stmt_elapsed_time - Most Recent Statement Elapsed Time" on page 384
	"stmt_node_number - Statement Node" on page 376
	"stmt_operation/operation - Statement Operation" on page 378
	"stmt_sorts - Statement Sorts" on page 385
	"stmt_start - Statement Operation Start Timestamp" on page 382
	"stmt_stop - Statement Operation Stop Timestamp" on page 383
	"stmt_sys_cpu_time - System CPU Time used by Statement" on page 405
	"stmt_text - SQL Statement Text monitor element" on page 384
	"stmt_type - Statement Type" on page 377
	"stmt_usr_cpu_time - User CPU Time used by Statement" on page 405
	"total_sort_time - Total Sort Time" on page 227

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 458
	"host_response_time - Host Response Time" on page 459
	"max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 445
	"max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 446
	"max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 447
	"max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 448
	"max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 449
	"max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 450
	"max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 450
	"max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 451
	"max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element" on page 452
	"max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element" on page 453
	"max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 454
	"max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 445
	"max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 446
	"max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 447
	"max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 447
	"max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 448
	"max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 449
	"max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 450
	"max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 451
	"max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 452
	"max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 453
	"max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 453

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

Snapshot logical data groups	Monitor element
stmt_transmissions (continued)	"max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms" on page 454
	"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms" on page 455
	"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms" on page 455
	"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms" on page 456
	"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms" on page 456
	"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms" on page 457
	"network_time_top - Maximum Network Time for Statement" on page 457
	"network_time_bottom - Minimum Network Time for Statement" on page 457
	"outbound_bytes_received - Outbound Number of Bytes Received" on page 442
	"outbound_bytes_sent - Outbound Number of Bytes Sent" on page 442
	"outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent" on page 443
	"outbound_bytes_received_top - Maximum Outbound Number of Bytes Received" on page 444
	"outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent" on page 444
	"outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received" on page 445
	"sql_chains - Number of SQL Chains Attempted" on page 438
"sql_stmts - Number of SQL Statements Attempted" on page 437	
subsection	"rows_read - Rows Read" on page 352
	"rows_written - Rows Written" on page 351
	"ss_exec_time - Subsection Execution Elapsed Time" on page 397
	"ss_node_number - Subsection Node Number" on page 396
	"ss_number - Subsection Number" on page 396
	"ss_status - Subsection Status" on page 397
	"ss_sys_cpu_time - System CPU Time used by Subsection" on page 407
	"ss_usr_cpu_time - User CPU Time used by Subsection" on page 407
	"tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed" on page 399
	"tq_id_waiting_on - Waited on Node on a Tablequeue" on page 401
	"tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows" on page 400
	"tq_node_waited_for - Waited for Node on a Tablequeue" on page 398
	"tq_rows_read - Number of Rows Read from Tablequeues" on page 399
	"tq_rows_written - Number of Rows Written to Tablequeues" on page 400
	"tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed" on page 398
"tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue" on page 398	

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 356
	"data_partition_id - Data partition identifier monitor element" on page 302
	"index_object_pages - Index Object Pages" on page 357
	"lob_object_pages - LOB Object Pages" on page 357
	"long_object_pages - Long Object Pages" on page 358
	"overflow_accesses - Accesses to Overflowed Records" on page 353
	"page_reorgs - Page Reorganizations" on page 356
	"rows_read - Rows Read" on page 352
	"rows_written - Rows Written" on page 351
	"table_file_id - Table File ID" on page 355
	"table_name - Table Name" on page 347
	"table_schema - Table Schema Name" on page 348
	"tablespace_id - Table Space Identification" on page 324
	"data_partition_id - Data partition identifier monitor element" on page 302
	"table_type - Table Type" on page 346
"xda_object_pages - XDA Object Pages" on page 358	
table_list	"db_conn_time - Database Activation Timestamp" on page 178
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
	"input_db_alias - Input Database Alias" on page 409
	"last_reset - Last Reset Timestamp" on page 409
table_reorg	"data_partition_id - Data partition identifier monitor element" on page 302
	"reorg_completion - Reorganization Completion Flag" on page 362
	"reorg_current_counter - Reorganize Progress" on page 361
	"reorg_end - Table Reorganize End Time" on page 362
	"reorg_index_id - Index Used to Reorganize the Table" on page 363
	"reorg_max_counter - Total Amount of Reorganization" on page 361
	"reorg_max_phase - Maximum Reorganize Phase" on page 361
	"reorg_phase - Reorganize Phase" on page 360
	"reorg_phase_start - Reorganize Phase Start Time" on page 360
	"reorg_start - Table Reorganize Start Time" on page 362
	"reorg_status - Table Reorganize Status" on page 359
	"reorg_tbsp_id - Table Space Where Table or Data partition is Reorganized" on page 363
	"reorg_type - Table Reorganize Attributes" on page 359
	"reorg_rows_compressed - Rows Compressed" on page 363
"reorg_rows_rejected_for_compression - Rows Rejected for Compression" on page 364	

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 274
	"direct_read_time - Direct Read Time" on page 275
	"direct_reads - Direct Reads From Database" on page 273
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_write_time - Direct Write Time" on page 276
	"direct_writes - Direct Writes to Database" on page 273
	"files_closed - Database Files Closed" on page 256
	"fs_caching - File System Caching" on page 335
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 263
	"pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 257
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 258
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 263
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 259
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 258
	"pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 261
	"pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 262
	"pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests" on page 264
	"pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads" on page 260
	"pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes" on page 261
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_data_writes - Buffer Pool Data Writes" on page 245
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_index_writes - Buffer Pool Index Writes" on page 249
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 266
"pool_read_time - Total Buffer Pool Physical Read Time" on page 255	

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

Snapshot logical data groups	Monitor element
tablespace (continued)	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 252
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 253
	"pool_write_time - Total Buffer Pool Physical Write Time" on page 255
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 250
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 253
	"pool_xda_writes - Buffer Pool XDA Data Writes" on page 254
	"tablespace_auto_resize_enabled - Auto-resize enabled" on page 336
	"tablespace_content_type - Table Space Contents Type" on page 326
	"tablespace_cur_pool_id - Buffer Pool Currently Being Used" on page 328
	"tablespace_extent_size - Table Space Extent Size" on page 328
	"tablespace_id - Table Space Identification" on page 324
	"tablespace_name - Table Space Name" on page 325
	"tablespace_next_pool_id - Buffer Pool That Will Be Used at Next Startup" on page 329
	"tablespace_page_size - Table Space Page Size" on page 327
	"tablespace_prefetch_size - Table Space Prefetch Size" on page 328
	"tablespace_rebalancer_mode - Rebalancer Mode" on page 331
"tablespace_type - Table Space Type" on page 326	
"tablespace_using_auto_storage - Using automatic storage" on page 336	
tablespace_container	"container_accessible - Accessibility of Container" on page 343
	"container_id - Container Identification" on page 341
	"container_name - Container Name" on page 341
	"container_stripe_set - Stripe Set" on page 342
	"container_total_pages - Total Pages in Container" on page 342
	"container_type - Container Type" on page 341
tablespace_list	"container_usable_pages - Usable Pages in Container" on page 342
	"db_conn_time - Database Activation Timestamp" on page 178
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
	"input_db_alias - Input Database Alias" on page 409
"last_reset - Last Reset Timestamp" on page 409	

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 337
	"tablespace_free_pages - Free Pages in Table Space" on page 330
	"tablespace_increase_size - Increase size in bytes" on page 337
	"tablespace_increase_size_percent - Increase size by percent" on page 338
	"tablespace_initial_size - Initial table space size" on page 336
	"tablespace_last_resize_failed - Last resize attempt failed" on page 339
	"tablespace_last_resize_time - Time of last successful resize" on page 338
	"tablespace_max_size - Maximum table space size" on page 337
	"tablespace_min_recovery_time - Minimum Recovery Time For Rollforward" on page 334
	"tablespace_num_containers - Number of Containers in Table Space" on page 335
	"tablespace_num_quiescers - Number of Quiescers" on page 333
	"tablespace_num_ranges - Number of Ranges in the Table Space Map" on page 335
	"tablespace_page_top - Table space high watermark" on page 331
	"tablespace_pending_free_pages - Pending Free Pages in Table Space" on page 330
	"tablespace_prefetch_size - Table Space Prefetch Size" on page 328
	"tablespace_rebalancer_extents_processed - Number of Extents the Rebalancer has Processed" on page 332
	"tablespace_rebalancer_extents_remaining - Total Number of Extents to be Processed by the Rebalancer" on page 332
	"tablespace_rebalancer_last_extent_moved - Last Extent Moved by the Rebalancer" on page 333
	"tablespace_rebalancer_priority - Current Rebalancer Priority" on page 333
	"tablespace_rebalancer_restart_time - Rebalancer Restart Time" on page 332
	"tablespace_rebalancer_start_time - Rebalancer Start Time" on page 331
	"tablespace_state - Table Space State" on page 326
	"tablespace_state_change_object_id - State Change Object Identification" on page 334
	"tablespace_state_change_ts_id - State Change Table Space Identification" on page 334
	"tablespace_total_pages - Total Pages in Table Space" on page 329
	"tablespace_usable_pages - Usable Pages in Table Space" on page 329
"tablespace_used_pages - Used Pages in Table Space" on page 330	
tablespace_quiescer	"quiescer_agent_id - Quiescer Agent Identification" on page 339
	"quiescer_auth_id - Quiescer User Authorization Identification" on page 339
	"quiescer_obj_id - Quiescer Object Identification" on page 340
	"quiescer_state - Quiescer State" on page 340
	"quiescer_ts_id - Quiescer Table Space Identification" on page 340

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 345</p> <p>“range_container_id - Range Container” on page 346</p> <p>“range_end_stripe - End Stripe” on page 345</p> <p>“range_max_extent - Maximum Extent in Range” on page 344</p> <p>“range_max_page_number - Maximum Page in Range” on page 344</p> <p>“range_num_containers - Number of Containers in Range” on page 345</p> <p>“range_number - Range Number” on page 344</p> <p>“range_offset - Range Offset” on page 346</p> <p>“range_start_stripe - Start Stripe” on page 345</p> <p>“range_stripe_set_number - Stripe Set Number” on page 344</p>
utility_info	<p>“utility_dbname - Database Operated on by Utility” on page 416</p> <p>“utility_id - Utility ID” on page 416</p> <p>“utility_invoker_type - Utility Invoker Type” on page 418</p> <p>“utility_state - Utility State” on page 418</p> <p>“utility_type - Utility Type” on page 417</p> <p>“utility_priority - Utility Priority” on page 417</p> <p>“utility_start_time - Utility Start Time” on page 417</p> <p>“utility_description - Utility Description” on page 417</p> <p>“node_number - Node Number” on page 202</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 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	"act_exec_time - Activity execution time monitor element" on page 476
	"activate_timestamp - Activate timestamp monitor element" on page 474
	"activity_id - Activity ID monitor element" on page 475
	"activity_secondary_id - Activity secondary ID monitor element" on page 475
	"activity_type - Activity type monitor element" on page 476
	"agent_id - Application Handle (agent ID)" on page 184
	"appl_id - Application ID" on page 189
	"appl_name - Application Name" on page 189
	"arm_correlator - Application response measurement correlator monitor element" on page 477
	"coord_partition_num - Coordinator partition number monitor element" on page 482
	"db_work_action_set_id - Database work action set ID monitor element" on page 487
	"db_work_class_id - Database work class ID monitor element" on page 487
	"parent_activity_id - Parent activity ID monitor element" on page 490
	"parent_uow_id - Parent unit of work ID monitor element" on page 490
	"partial_record - Partial Record monitor element" on page 412
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"sc_work_action_set_id - Service class work action set ID monitor element" on page 494
	"sc_work_class_id - Service class work class ID monitor element" on page 494
	"service_subclass_name - Service subclass name monitor element" on page 495
	"service_superclass_name - Service superclass name monitor element" on page 496
	"session_auth_id - Session Authorization ID" on page 193
	"sort_overflows - Sort Overflows" on page 228
	"sqlca - SQL Communications Area (SQLCA)" on page 387
	"time_completed - Time completed monitor element" on page 500
	"time_created - Time created monitor element" on page 501
	"time_started - Time started monitor element" on page 501
	"total_sort_time - Total Sort Time" on page 227
	"total_sorts - Total Sorts" on page 227
	"tpmon_acc_str - TP Monitor Client Accounting String" on page 464
	"tpmon_client_app - TP Monitor Client Application Name" on page 464
	"tpmon_client_userid - TP Monitor Client User ID" on page 463
	"tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 463
	"uow_id - Unit of work ID monitor element" on page 502
	"workload_id - Workload ID monitor element" on page 504
	"workload_occurrence_id - Workload occurrence identifier monitor element" on page 505

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 246	
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248	
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242	
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244	
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247	
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249	
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads" on page 252	
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads" on page 253	
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads" on page 250	
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads" on page 253	
	"prep_time - Preparation time monitor element" on page 491	
	"query_card_estimate - Query Number of Rows Estimate" on page 387	
	"query_cost_estimate - Query Cost Estimate" on page 388	
	"rows_fetched - Rows fetched monitor element" on page 492	
	"rows_modified - Rows modified monitor element" on page 492	
	"rows_returned - Rows returned monitor element" on page 493	
	"system_cpu_time - System CPU Time" on page 406	
	"user_cpu_time - User CPU Time" on page 406	
	event_activitystmt	"activate_timestamp - Activate timestamp monitor element" on page 474
		"activity_id - Activity ID monitor element" on page 475
"activity_secondary_id - Activity secondary ID monitor element" on page 475		
"appl_id - Application ID" on page 189		
"comp_env_desc - Compilation environment handle" on page 393		
"creator - Application Creator" on page 382		
"package_name - Package Name" on page 379		
"package_version_id - Package Version" on page 380		
"section_env - Section environment monitor element" on page 495		
"section_number - Section Number" on page 381		
"stmt_first_use_time - Statement first use time" on page 389		
"stmt_isolation - Statement isolation" on page 390		
"stmt_last_use_time - Statement last use time monitor element" on page 389		
"stmt_lock_timeout - Statement lock timeout" on page 390		
"stmt_nest_level - Statement nesting level" on page 391		
"stmt_pkgcache_id - Statement package cache identifier" on page 393		
"stmt_query_id - Statement query identifier" on page 392		
"stmt_source_id - Statement source identifier" on page 392		
"stmt_text - SQL Statement Text monitor element" on page 384		
"stmt_type - Statement Type" on page 377		
"uow_id - Unit of work ID monitor element" on page 502		

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 474
	"activity_id - Activity ID monitor element" on page 475
	"activity_secondary_id - Activity secondary ID monitor element" on page 475
	"appl_id - Application ID" on page 189
	"stmt_value_data - Value data" on page 394
	"stmt_value_index - Value index" on page 395
	"stmt_value_isnull - Value has null value" on page 394
	"stmt_value_isreopt - Variable used for statement reoptimization" on page 395
	"stmt_value_type - Value type" on page 394
	"uow_id - Unit of work ID monitor element" on page 502
event_bufferpool	"bp_id - Buffer pool identifier monitor element" on page 241
	"bp_name - Buffer Pool Name" on page 268
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
	"direct_read_reqs - Direct Read Requests" on page 274
	"direct_read_time - Direct Read Time" on page 275
	"direct_reads - Direct Reads From Database" on page 273
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_write_time - Direct Write Time" on page 276
	"direct_writes - Direct Writes to Database" on page 273
	"event_time - Event Time" on page 413
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"evmon_flushes - Number of Event Monitor Flushes" on page 414
	"files_closed - Database Files Closed" on page 256
	"partial_record - Partial Record monitor element" on page 412
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 263
	"pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 257
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 258
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 259
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 258
	"pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 261
	"pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 262
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_data_writes - Buffer Pool Data Writes" on page 245
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_index_writes - Buffer Pool Index Writes" on page 249
	"pool_read_time - Total Buffer Pool Physical Read Time" on page 255
	"pool_write_time - Total Buffer Pool Physical Write Time" on page 255

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 366
	"agent_id - Application Handle (agent ID)" on page 184
	"appl_id - Application ID" on page 189
	"appl_priority - Application Agent Priority" on page 199
	"appl_priority_type - Application Priority Type" on page 200
	"appl_section_inserts - Section Inserts monitor element" on page 289
	"appl_section_lookups - Section Lookups" on page 289
	"authority_bitmap - User Authorization Level monitor element" on page 201
	"authority_lvl - User Authorization Level" on page 200
	"binds_precompiles - Binds/Precompiles Attempted" on page 376
	"cat_cache_inserts - Catalog Cache Inserts" on page 278
	"cat_cache_lookups - Catalog Cache Lookups" on page 277
	"cat_cache_overflows - Catalog Cache Overflows" on page 278
	"commit_sql_stmts - Commit Statements Attempted" on page 369
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 372
	"deadlocks - Deadlocks Detected" on page 303
	"direct_read_reqs - Direct Read Requests" on page 274
	"direct_read_time - Direct Read Time" on page 275
	"direct_reads - Direct Reads From Database" on page 273
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_write_time - Direct Write Time" on page 276
	"direct_writes - Direct Writes to Database" on page 273
	"disconn_time - Database Deactivation Timestamp" on page 178
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 368
	"failed_sql_stmts - Failed Statement Operations" on page 368
	"hash_join_overflows - Hash Join Overflows" on page 233
	"hash_join_small_overflows - Hash Join Small Overflows" on page 234
	"int_auto_rebinds - Internal Automatic Rebinds" on page 373
	"int_commits - Internal Commits" on page 373
	"int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock" on page 375
	"int_rollbacks - Internal Rollbacks" on page 374
	"int_rows_deleted - Internal Rows Deleted" on page 353
	"int_rows_inserted - Internal Rows Inserted" on page 355
	"int_rows_updated - Internal Rows Updated" on page 354
	"lock_escalation - Lock Escalation" on page 310
	"lock_timeouts - Number of Lock Timeouts" on page 309
	"lock_wait_time - Time Waited On Locks" on page 318
	"lock_waits - Lock Waits" on page 317
	"olap_func_overflows - OLAP Function Overflows monitor element" on page 235
	"partial_record - Partial Record monitor element" on page 412

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

Event logical data groups	Monitor element name
event_conn (continued)	<p>"int_rows_updated - Internal Rows Updated" on page 354</p> <p>"pkg_cache_inserts - Package Cache Inserts" on page 281</p> <p>"pkg_cache_lookups - Package Cache Lookups" on page 280</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243</p> <p>"pool_data_writes - Buffer Pool Data Writes" on page 245</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248</p> <p>"pool_index_writes - Buffer Pool Index Writes" on page 249</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time" on page 255</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244</p> <p>"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247</p> <p>"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249</p> <p>"pool_write_time - Total Buffer Pool Physical Write Time" on page 255</p> <p>"prefetch_wait_time - Time Waited for Prefetch" on page 268</p> <p>"priv_workspace_num_overflows - Private Workspace Overflows" on page 287</p> <p>"priv_workspace_section_inserts - Private Workspace Section Inserts" on page 288</p> <p>"priv_workspace_section_lookups - Private Workspace Section Lookups" on page 287</p> <p>"priv_workspace_size_top - Maximum Private Workspace Size" on page 286</p> <p>"rej_curs_blk - Rejected Block Cursor Requests" on page 365</p> <p>"rollback_sql_stmts - Rollback Statements Attempted" on page 370</p> <p>"rows_read - Rows Read" on page 352</p> <p>"rows_selected - Rows Selected" on page 350</p> <p>"rows_written - Rows Written" on page 351</p> <p>"select_sql_stmts - Select SQL Statements Executed" on page 371</p> <p>"sequence_no - Sequence number monitor element" on page 191</p> <p>"shr_workspace_num_overflows - Shared Workspace Overflows" on page 284</p> <p>"shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 285</p> <p>"shr_workspace_section_lookups - Shared Workspace Section Lookups" on page 285</p> <p>"shr_workspace_size_top - Maximum Shared Workspace Size" on page 283</p> <p>"sort_overflows - Sort Overflows" on page 228</p> <p>"static_sql_stmts - Static SQL Statements Attempted" on page 367</p> <p>"system_cpu_time - System CPU Time" on page 406</p> <p>"total_hash_joins - Total Hash Joins" on page 231</p> <p>"total_hash_loops - Total Hash Loops" on page 233</p> <p>"total_olap_funcs - Total OLAP Functions monitor element" on page 235</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 219
	"total_sort_time - Total Sort Time" on page 227
	"total_sorts - Total Sorts" on page 227
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 371
	"unread_prefetch_pages - Unread Prefetch Pages" on page 269
	"user_cpu_time - User CPU Time" on page 406
	"x_lock_escals - Exclusive Lock Escalations" on page 305
	"xquery_stmts - XQuery Statements Attempted" on page 377
event_connheader	"agent_id - Application Handle (agent ID)" on page 184
	"appl_id - Application ID" on page 189
	"appl_name - Application Name" on page 189
	"auth_id - Authorization ID" on page 192
	"client_db_alias - Database Alias Used by Application" on page 194
	"client_pid - Client Process ID" on page 197
	"client_platform - Client Operating Platform" on page 198
	"client_prdid - Client Product/Version ID" on page 193
	"client_protocol - Client Communication Protocol" on page 198
	"codepage_id - ID of Code Page Used by Application" on page 187
	"conn_time - Time of Database Connection" on page 178
	"corr_token - DRDA Correlation Token" on page 197
	"execution_id - User Login ID" on page 196
	"node_number - Node Number" on page 202
	"sequence_no - Sequence number monitor element" on page 191
"territory_code - Database Territory Code" on page 199	
event_connmemuse	"node_number - Node Number" on page 202
	"pool_config_size - Configured Size of Memory Pool" on page 222
	"pool_cur_size - Current Size of Memory Pool" on page 222
	"pool_id - Memory Pool Identifier" on page 220
	"pool_secondary_id - Memory Pool Secondary Identifier" on page 221
	"pool_watermark - Memory Pool Watermark" on page 223
event_data_value	"deadlock_id - Deadlock Event Identifier" on page 311
	"deadlock_node - Partition Number Where Deadlock Occurred" on page 312
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"participant_no - Participant within Deadlock" on page 312
	"stmt_history_id - Statement history identifier" on page 389
	"stmt_value_data - Value data" on page 394
	"stmt_value_index - Value index" on page 395
	"stmt_value_isnull - Value has null value" on page 394
	"stmt_value_isreopt - Variable used for statement reoptimization" on page 395
"stmt_value_type - Value type" on page 394	

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 233
	"appl_section_inserts - Section Inserts monitor element" on page 289
	"appl_section_lookups - Section Lookups" on page 289
	"async_runstats - Total number of asynchronous RUNSTATS requests monitor element" on page 507
	"binds_precompiles - Binds/Precompiles Attempted" on page 376
	"blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element" on page 301
	"cat_cache_inserts - Catalog Cache Inserts" on page 278
	"cat_cache_lookups - Catalog Cache Lookups" on page 277
	"cat_cache_overflows - Catalog Cache Overflows" on page 278
	"cat_cache_size_top - Catalog cache high watermark" on page 279
	"catalog_node - Catalog Node Number" on page 180
	"catalog_node_name - Catalog Node Network Name" on page 179
	"commit_sql_stmts - Commit Statements Attempted" on page 369
	"connections_top - Maximum Number of Concurrent Connections" on page 204
	"db_heap_top - Maximum Database Heap Allocated" on page 290
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements" on page 372
	"deadlocks - Deadlocks Detected" on page 303
	"direct_read_reqs - Direct Read Requests" on page 274
	"direct_read_time - Direct Read Time" on page 275
	"direct_reads - Direct Reads From Database" on page 273
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_write_time - Direct Write Time" on page 276
	"direct_writes - Direct Writes to Database" on page 273
	"disconn_time - Database Deactivation Timestamp" on page 178
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted" on page 368
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"evmon_flushes - Number of Event Monitor Flushes" on page 414
	"failed_sql_stmts - Failed Statement Operations" on page 368
	"files_closed - Database Files Closed" on page 256
	"hash_join_overflows - Hash Join Overflows" on page 233
	"hash_join_small_overflows - Hash Join Small Overflows" on page 234
	"int_auto_rebinds - Internal Automatic Rebinds" on page 373
	"int_commits - Internal Commits" on page 373
	"int_rollbacks - Internal Rollbacks" on page 374
	"int_rows_deleted - Internal Rows Deleted" on page 353
	"int_rows_inserted - Internal Rows Inserted" on page 355
	"int_rows_updated - Internal Rows Updated" on page 354
	"lock_escals - Number of Lock Escalations" on page 304
	"lock_timeouts - Number of Lock Timeouts" on page 309
	"lock_wait_time - Time Waited On Locks" on page 318
	"lock_waits - Lock Waits" on page 317
	"log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages" on page 295

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 296</p> <p>"log_reads - Number of Log Pages Read" on page 292</p> <p>"log_to_redo_for_recovery - Amount of Log to be Redone for Recovery" on page 296</p> <p>"log_write_time - Log Write Time" on page 296</p> <p>"log_writes - Number of Log Pages Written" on page 293</p> <p>"num_log_read_io - Number of Log Reads" on page 297</p> <p>"num_log_write_io - Number of Log Writes" on page 297</p> <p>"num_threshold_violations - Number of threshold violations monitor element" on page 489</p> <p>"olap_func_overflows - OLAP Function Overflows monitor element" on page 235</p> <p>"partial_record - Partial Record monitor element" on page 412</p> <p>"pkg_cache_inserts - Package Cache Inserts" on page 281</p> <p>"pkg_cache_lookups - Package Cache Lookups" on page 280</p> <p>"pkg_cache_num_overflows - Package Cache Overflows" on page 282</p> <p>"pkg_cache_size_top - Package cache high watermark" on page 282</p> <p>"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 263</p> <p>"pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 257</p> <p>"pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 258</p> <p>"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 263</p> <p>"pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 259</p> <p>"pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 258</p> <p>"pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 261</p> <p>"pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 262</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243</p> <p>"pool_data_writes - Buffer Pool Data Writes" on page 245</p> <p>"pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered" on page 265</p> <p>"pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered" on page 267</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248</p> <p>"pool_index_writes - Buffer Pool Index Writes" on page 249</p> <p>"pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered" on page 265</p> <p>"pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 266</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time" on page 255</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244</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 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"pool_write_time - Total Buffer Pool Physical Write Time" on page 255
	"post_shrthreshold_hash_joins - Post threshold hash joins" on page 232
	"post_shrthreshold_sorts - Post shared threshold sorts" on page 225
	"prefetch_wait_time - Time Waited for Prefetch" on page 268
	"priv_workspace_num_overflows - Private Workspace Overflows" on page 287
	"priv_workspace_section_inserts - Private Workspace Section Inserts" on page 288
	"priv_workspace_section_lookups - Private Workspace Section Lookups" on page 287
	"priv_workspace_size_top - Maximum Private Workspace Size" on page 286
	"rollback_sql_stmts - Rollback Statements Attempted" on page 370
	"rows_deleted - Rows Deleted" on page 349
	"rows_inserted - Rows Inserted" on page 349
	"rows_read - Rows Read" on page 352
	"rows_selected - Rows Selected" on page 350
	"rows_updated - Rows Updated" on page 350
	"sec_log_used_top - Maximum Secondary Log Space Used" on page 290
	"select_sql_stmts - Select SQL Statements Executed" on page 371
	"server_platform - Server Operating System" on page 175
	"shr_workspace_num_overflows - Shared Workspace Overflows" on page 284
	"shr_workspace_section_inserts - Shared Workspace Section Inserts" on page 285
	"shr_workspace_section_lookups - Shared Workspace Section Lookups" on page 285
	"shr_workspace_size_top - Maximum Shared Workspace Size" on page 283
	"sort_overflows - Sort Overflows" on page 228
	"static_sql_stmts - Static SQL Statements Attempted" on page 367
	"stats_cache_size - Size of statistics cache monitor element" on page 506
	"stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 508
	"stats_fabrications - Total number of statistics fabrications monitor elements" on page 506
	"sync_runstats - Total number of synchronous RUNSTATS activities monitor element" on page 507
	"sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 509
	"tot_log_used_top - Maximum Total Log Space Used" on page 291
	"total_cons - Connects Since Database Activation" on page 213
	"total_hash_joins - Total Hash Joins" on page 231
	"total_hash_loops - Total Hash Loops" on page 233
	"total_olap_funcs - Total OLAP Functions monitor element" on page 235

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 227
	"total_sorts - Total Sorts" on page 227
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed" on page 371
	"unread_prefetch_pages - Unread Prefetch Pages" on page 269
	"x_lock_escals - Exclusive Lock Escalations" on page 305
	"xquery_stmts - XQuery Statements Attempted" on page 377
event_dbheader	"conn_time - Time of Database Connection" on page 178
	"db_name - Database Name" on page 176
	"db_path - Database Path" on page 177
event_dbmemuse	"node_number - Node Number" on page 202
	"pool_config_size - Configured Size of Memory Pool" on page 222
	"pool_cur_size - Current Size of Memory Pool" on page 222
	"pool_id - Memory Pool Identifier" on page 220
	"pool_watermark - Memory Pool Watermark" on page 223
event_deadlock	"deadlock_id - Deadlock Event Identifier" on page 311
	"deadlock_node - Partition Number Where Deadlock Occurred" on page 312
	"dl_conns - Connections Involved in Deadlock" on page 310
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"rolled_back_agent_id - Rolled Back Agent" on page 322
	"rolled_back_appl_id - Rolled Back Application" on page 322
	"rolled_back_participant_no - Rolled Back Application Participant" on page 313
	"rolled_back_sequence_no - Rolled Back Sequence Number" on page 322
"start_time - Event Start Time" on page 383	

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 184
	"appl_id - Application ID" on page 189
	"appl_id_holding_lk - Application ID Holding Lock" on page 321
	"blocking_cursor - Blocking Cursor" on page 462
	"consistency_token - Package Consistency Token" on page 380
	"creator - Application Creator" on page 382
	"cursor_name - Cursor Name" on page 381
	"data_partition_id - Data partition identifier monitor element" on page 302
	"deadlock_id - Deadlock Event Identifier" on page 311
	"deadlock_node - Partition Number Where Deadlock Occurred" on page 312
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"lock_escalation - Lock Escalation" on page 310
	"lock_mode - Lock Mode" on page 306
	"lock_mode_requested - Lock Mode Requested" on page 311
	"lock_node - Lock Node" on page 309
	"lock_object_name - Lock Object Name" on page 308
	"lock_object_type - Lock Object Type Waited On" on page 307
	"lock_wait_start_time - Lock Wait Start Timestamp" on page 319
	"locks_held - Locks Held" on page 301
	"locks_in_list - Number of Locks Reported" on page 313
	"package_name - Package Name" on page 379
	"package_version_id - Package Version" on page 380
	"participant_no - Participant within Deadlock" on page 312
	"participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application" on page 313
	"section_number - Section Number" on page 381
	"sequence_no - Sequence number monitor element" on page 191
	"sequence_no_holding_lk - Sequence Number Holding Lock" on page 321
	"start_time - Event Start Time" on page 383
	"stmt_operation/operation - Statement Operation" on page 378
	"stmt_text - SQL Statement Text monitor element" on page 384
	"stmt_type - Statement Type" on page 377
	"table_name - Table Name" on page 347
	"table_schema - Table Schema Name" on page 348
	"tablespace_name - Table Space Name" on page 325

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 184
	"appl_id - Application ID" on page 189
	"appl_id_holding_lk - Application ID Holding Lock" on page 321
	"data_partition_id - Data partition identifier monitor element" on page 302
	"deadlock_id - Deadlock Event Identifier" on page 311
	"deadlock_node - Partition Number Where Deadlock Occurred" on page 312
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"lock_attributes - Lock Attributes" on page 314
	"lock_count - Lock Count" on page 315
	"lock_current_mode - Original Lock Mode Before Conversion" on page 316
	"lock_escalation - Lock Escalation" on page 310
	"lock_hold_count - Lock Hold Count" on page 316
	"lock_mode - Lock Mode" on page 306
	"lock_mode_requested - Lock Mode Requested" on page 311
	"lock_name - Lock Name" on page 313
	"lock_node - Lock Node" on page 309
	"lock_object_name - Lock Object Name" on page 308
	"lock_object_type - Lock Object Type Waited On" on page 307
	"lock_release_flags - Lock Release Flags" on page 315
	"lock_wait_start_time - Lock Wait Start Timestamp" on page 319
	"participant_no - Participant within Deadlock" on page 312
	"participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application" on page 313
	"sequence_no - Sequence number monitor element" on page 191
	"sequence_no_holding_lk - Sequence Number Holding Lock" on page 321
	"start_time - Event Start Time" on page 383
	"table_name - Table Name" on page 347
	"table_schema - Table Schema Name" on page 348
	"tablespace_name - Table Space Name" on page 325
	"tpmon_acc_str - TP Monitor Client Accounting String" on page 464
	"tpmon_client_app - TP Monitor Client Application Name" on page 464
"tpmon_client_userid - TP Monitor Client User ID" on page 463	
"tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 463	
event_histogrambin	"bin_id - Histogram bin identifier monitor element" on page 478
	"bottom - Histogram bin bottom monitor element" on page 478
	"histogram_type - Histogram type monitor element" on page 488
	"number_in_bin - Number in bin monitor element" on page 489
	"service_class_id - Service class ID monitor element" on page 495
	"statistics_timestamp - Statistics timestamp monitor element" on page 496
	"top - Histogram bin top monitor element" on page 502
	"work_action_set_id - Work action set ID monitor element" on page 503
"work_class_id - Work class ID monitor element" on page 504	

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 411
	"codepage_id - ID of Code Page Used by Application" on page 187
	"event_monitor_name - Event Monitor Name" on page 412
	"num_nodes_in_db2_instance - Number of Nodes in Partition" on page 410
	"server_instance_name - Server Instance Name" on page 172
	"server_prdid - Server Product/Version ID" on page 173
	"territory_code - Database Territory Code" on page 199
	"version - Version of Monitor Data" on page 412
event_overflow	"count - Number of Event Monitor Overflows" on page 410
	"first_overflow_time - Time of First Event Overflow" on page 411
	"last_overflow_time - Time of Last Event Overflow" on page 411
	"node_number - Node Number" on page 202
event_qstats	"last_wlm_reset - Time of last reset monitor element" on page 489
	"queue_assignments_total - Queue assignments total monitor element" on page 491
	"queue_size_top - Queue size top monitor element" on page 491
	"queue_time_total - Queue time total monitor element" on page 492
	"service_subclass_name - Service subclass name monitor element" on page 495
	"service_superclass_name - Service superclass name monitor element" on page 496
	"statistics_timestamp - Statistics timestamp monitor element" on page 496
	"threshold_domain - Threshold domain monitor element" on page 498
	"threshold_name - Threshold name monitor element" on page 499
	"threshold_predicate - Threshold predicate monitor element" on page 499
	"thresholdid - Threshold ID monitor element" on page 500
	"work_action_set_name - Work action set name monitor element" on page 503
"work_class_name - Work class name monitor element" on page 504	

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

Event logical data groups	Monitor element name
event_scstats	"concurrent_act_top - Concurrent activity top monitor element" on page 478
	"concurrent_connection_top - Concurrent connection top monitor element" on page 479
	"coord_act_aborted_total - Coordinator activities aborted total monitor element" on page 480
	"coord_act_completed_total - Coordinator activities completed total monitor element" on page 481
	"coord_act_est_cost_avg - Coordinator activity estimated cost average monitor element" on page 485
	"coord_act_exec_time_avg - Coordinator activities execution time average monitor element" on page 484
	"coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element" on page 486
	"coord_act_lifetime_avg - Coordinator activity lifetime average monitor element" on page 483
	"coord_act_lifetime_top - Coordinator activity lifetime top monitor element" on page 481
	"coord_act_queue_time_avg - Coordinator activity queue time average monitor element" on page 483
	"coord_act_rejected_total - Coordinator activities rejected total monitor element" on page 481
	"cost_estimate_top - Cost estimate top monitor element" on page 482
	"last_wlm_reset - Time of last reset monitor element" on page 489
	"request_exec_time_avg - Request execution time average monitor element" on page 485
	"rows_returned_top - Actual rows returned top monitor element" on page 493
	"service_class_id - Service class ID monitor element" on page 495
	"service_subclass_name - Service subclass name monitor element" on page 495
	"service_superclass_name - Service superclass name monitor element" on page 496
	"statistics_timestamp - Statistics timestamp monitor element" on page 496
	"temp_tablespace_top - Temporary table space top monitor element" on page 497
event_start	"start_time - Event Start Time" on page 383

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 184
	"agents_top - Number of Agents Created" on page 403
	"appl_id - Application ID" on page 189
	"blocking_cursor - Blocking Cursor" on page 462
	"consistency_token - Package Consistency Token" on page 380
	"creator - Application Creator" on page 382
	"cursor_name - Cursor Name" on page 381
	"fetch_count - Number of Successful Fetches" on page 386
	"int_rows_deleted - Internal Rows Deleted" on page 353
	"int_rows_inserted - Internal Rows Inserted" on page 355
	"int_rows_updated - Internal Rows Updated" on page 354
	"package_name - Package Name" on page 379
	"package_version_id - Package Version" on page 380
	"partial_record - Partial Record monitor element" on page 412
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"rows_read - Rows Read" on page 352
	"rows_written - Rows Written" on page 351
	"section_number - Section Number" on page 381
	"sequence_no - Sequence number monitor element" on page 191
	"sort_overflows - Sort Overflows" on page 228
	"sql_req_id - Request Identifier for SQL Statement" on page 415
	"sqlca - SQL Communications Area (SQLCA)" on page 387
	"start_time - Event Start Time" on page 383
	"stats_fabricate_time - Total time spent on statistics fabrication activities monitor element" on page 508
	"stmt_operation/operation - Statement Operation" on page 378
	"stmt_text - SQL Statement Text monitor element" on page 384
	"stmt_type - Statement Type" on page 377
	"stop_time - Event Stop Time" on page 383
	"sync_runstats_time - Total time spent on synchronous RUNSTATS activities monitor element" on page 509
	"system_cpu_time - System CPU Time" on page 406
	"total_sort_time - Total Sort Time" on page 227
	"total_sorts - Total Sorts" on page 227
	"user_cpu_time - User CPU Time" on page 406

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

Event logical data groups	Monitor element name
event_stmt_history	"comp_env_desc - Compilation environment handle" on page 393
	"creator - Application Creator" on page 382
	"deadlock_id - Deadlock Event Identifier" on page 311
	"deadlock_node - Partition Number Where Deadlock Occurred" on page 312
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"package_name - Package Name" on page 379
	"package_version_id - Package Version" on page 380
	"participant_no - Participant within Deadlock" on page 312
	"section_number - Section Number" on page 381
	"sequence_no - Sequence number monitor element" on page 191
	"stmt_first_use_time - Statement first use time" on page 389
	"stmt_history_id - Statement history identifier" on page 389
	"stmt_invocation_id - Statement invocation identifier" on page 391
	"stmt_isolation - Statement isolation" on page 390
	"stmt_last_use_time - Statement last use time monitor element" on page 389
	"stmt_lock_timeout - Statement lock timeout" on page 390
	"stmt_nest_level - Statement nesting level" on page 391
	"stmt_pkgcache_id - Statement package cache identifier" on page 393
	"stmt_query_id - Statement query identifier" on page 392
	"stmt_source_id - Statement source identifier" on page 392
	"stmt_text - SQL Statement Text monitor element" on page 384
"stmt_type - Statement Type" on page 377	
event_subsection	"agent_id - Application Handle (agent ID)" on page 184
	"num_agents - Number of Agents Working on a Statement" on page 403
	"partial_record - Partial Record monitor element" on page 412
	"ss_exec_time - Subsection Execution Elapsed Time" on page 397
	"ss_node_number - Subsection Node Number" on page 396
	"ss_number - Subsection Number" on page 396
	"ss_sys_cpu_time - System CPU Time used by Subsection" on page 407
	"ss_usr_cpu_time - User CPU Time used by Subsection" on page 407
	"tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows" on page 400
	"tq_rows_read - Number of Rows Read from Tablequeues" on page 399
	"tq_rows_written - Number of Rows Written to Tablequeues" on page 400
	"tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed" on page 398

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 356</p> <p>“data_partition_id - Data partition identifier monitor element” on page 302</p> <p>“event_time - Event Time” on page 413</p> <p>“evmon_activates - Number of Event Monitor Activations” on page 414</p> <p>“evmon_flushes - Number of Event Monitor Flushes” on page 414</p> <p>“index_object_pages - Index Object Pages” on page 357</p> <p>“lob_object_pages - LOB Object Pages” on page 357</p> <p>“long_object_pages - Long Object Pages” on page 358</p> <p>“overflow_accesses - Accesses to Overflowed Records” on page 353</p> <p>“page_reorgs - Page Reorganizations” on page 356</p> <p>“partial_record - Partial Record monitor element” on page 412</p> <p>“rows_read - Rows Read” on page 352</p> <p>“rows_written - Rows Written” on page 351</p> <p>“table_name - Table Name” on page 347</p> <p>“table_schema - Table Schema Name” on page 348</p> <p>“table_type - Table Type” on page 346</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 274
	"direct_read_time - Direct Read Time" on page 275
	"direct_reads - Direct Reads From Database" on page 273
	"direct_write_reqs - Direct Write Requests" on page 275
	"direct_write_time - Direct Write Time" on page 276
	"direct_writes - Direct Writes to Database" on page 273
	"event_time - Event Time" on page 413
	"evmon_activates - Number of Event Monitor Activations" on page 414
	"evmon_flushes - Number of Event Monitor Flushes" on page 414
	"files_closed - Database Files Closed" on page 256
	"partial_record - Partial Record monitor element" on page 412
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests" on page 263
	"pool_async_data_reads - Buffer pool asynchronous data reads monitor element" on page 257
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes" on page 258
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests" on page 263
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads" on page 259
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes" on page 258
	"pool_async_read_time - Buffer Pool Asynchronous Read Time" on page 261
	"pool_async_write_time - Buffer Pool Asynchronous Write Time" on page 262
	"pool_data_l_reads - Buffer Pool Data Logical Reads" on page 241
	"pool_data_p_reads - Buffer Pool Data Physical Reads" on page 243
	"pool_data_writes - Buffer Pool Data Writes" on page 245
	"pool_index_l_reads - Buffer Pool Index Logical Reads" on page 246
	"pool_index_p_reads - Buffer Pool Index Physical Reads" on page 248
	"pool_index_writes - Buffer Pool Index Writes" on page 249
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers" on page 266
	"pool_read_time - Total Buffer Pool Physical Read Time" on page 255
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads" on page 242
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads" on page 244
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads" on page 247
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads" on page 249
	"pool_write_time - Total Buffer Pool Physical Write Time" on page 255
	"tablespace_name - Table Space Name" on page 325

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

Event logical data groups	Monitor element name
event_thresholdviolations	"activate_timestamp - Activate timestamp monitor element" on page 474
	"activity_collected - Activity collected monitor element" on page 474
	"activity_id - Activity ID monitor element" on page 475
	"agent_id - Application Handle (agent ID)" on page 184
	"appl_id - Application ID" on page 189
	"coord_partition_num - Coordinator partition number monitor element" on page 482
	"threshold_action - Threshold action monitor element" on page 497
	"threshold_maxvalue - Threshold maximum value monitor element" on page 498
	"threshold_predicate - Threshold predicate monitor element" on page 499
	"threshold_queuesize - Threshold queue size monitor element" on page 499
	"thresholdid - Threshold ID monitor element" on page 500
	"time_of_violation - Time of violation monitor element" on page 501
	"uow_id - Unit of work ID monitor element" on page 502
event_wlstats	"concurrent_wlo_act_top - Concurrent WLO activity top monitor element" on page 479
	"concurrent_wlo_top - Concurrent workload occurrences top monitor element" on page 480
	"coord_act_aborted_total - Coordinator activities aborted total monitor element" on page 480
	"coord_act_completed_total - Coordinator activities completed total monitor element" on page 481
	"coord_act_rejected_total - Coordinator activities rejected total monitor element" on page 481
	"last_wlm_reset - Time of last reset monitor element" on page 489
	"statistics_timestamp - Statistics timestamp monitor element" on page 496
	"wlo_completed_total - Workload occurrences completed total monitor element" on page 502
	"workload_id - Workload ID monitor element" on page 504
	"workload_name - Workload name monitor element" on page 505

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 477
	"coord_act_est_cost_avg - Coordinator activity estimated cost average monitor element" on page 485
	"coord_act_exec_time_avg - Coordinator activities execution time average monitor element" on page 484
	"coord_act_interarrival_time_avg - Coordinator activity arrival time average monitor element" on page 486
	"coord_act_lifetime_avg - Coordinator activity lifetime average monitor element" on page 483
	"coord_act_lifetime_top - Coordinator activity lifetime top monitor element" on page 481
	"coord_act_queue_time_avg - Coordinator activity queue time average monitor element" on page 483
	"cost_estimate_top - Cost estimate top monitor element" on page 482
	"last_wlm_reset - Time of last reset monitor element" on page 489
	"rows_returned_top - Actual rows returned top monitor element" on page 493
	"statistics_timestamp - Statistics timestamp monitor element" on page 496
	"temp_tablespace_top - Temporary table space top monitor element" on page 497
	"work_action_set_id - Work action set ID monitor element" on page 503
	"work_action_set_name - Work action set name monitor element" on page 503
	"work_class_id - Work class ID monitor element" on page 504
	"work_class_name - Work class name monitor element" on page 504
	event_xact
"appl_id - Application ID" on page 189	
"lock_escals - Number of Lock Escalations" on page 304	
"lock_wait_time - Time Waited On Locks" on page 318	
"locks_held_top - Maximum Number of Locks Held" on page 310	
"partial_record - Partial Record monitor element" on page 412	
"prev_uow_stop_time - Previous Unit of Work Completion Timestamp" on page 205	
"rows_read - Rows Read" on page 352	
"rows_written - Rows Written" on page 351	
"sequence_no - Sequence number monitor element" on page 191	
"system_cpu_time - System CPU Time" on page 406	
"tpmon_acc_str - TP Monitor Client Accounting String" on page 464	
"tpmon_client_app - TP Monitor Client Application Name" on page 464	
"tpmon_client_userid - TP Monitor Client User ID" on page 463	
"tpmon_client_wkstn - TP Monitor Client Workstation Name" on page 463	
"uow_log_space_used - Unit of Work Log Space Used" on page 293	
"uow_start_time - Unit of Work Start Timestamp" on page 205	
"uow_status - Unit of Work Status" on page 207	
"uow_stop_time - Unit of Work Stop Timestamp" on page 206	
"user_cpu_time - User CPU Time" on page 406	
"x_lock_escals - Exclusive Lock Escalations" on page 305	

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

Event logical data groups	Monitor element name
lock	"data_partition_id - Data partition identifier monitor element" on page 302
	"lock_attributes - Lock Attributes" on page 314
	"lock_count - Lock Count" on page 315
	"lock_current_mode - Original Lock Mode Before Conversion" on page 316
	"lock_escalation - Lock Escalation" on page 310
	"lock_hold_count - Lock Hold Count" on page 316
	"lock_mode - Lock Mode" on page 306
	"lock_name - Lock Name" on page 313
	"lock_object_name - Lock Object Name" on page 308
	"lock_object_type - Lock Object Type Waited On" on page 307
	"lock_release_flags - Lock Release Flags" on page 315
	"lock_status - Lock Status" on page 307
	"node_number - Node Number" on page 202
	"table_file_id - Table File ID" on page 355
"table_name - Table Name" on page 347	
"table_schema - Table Schema Name" on page 348	
"tablespace_name - Table Space Name" on page 325	
sqlca	sqlcabc
	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 event_activitystmt
WITH DETAILS AND VALUES	event_activity event_activitystmt 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 not 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	dcs_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	Database Connect Pending: The application has initiated a database connection but the request has not yet completed.
SQLM_CONNECTED	Database Connect Completed: The application has initiated a database connection and the request has completed.
SQLM_UOWEXEC	Unit of Work Executing: The database manager is executing requests on behalf of the unit of work.
SQLM_UOWWAIT	Unit of Work waiting: 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	Lock Wait: 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	Commit Active: The unit of work is committing its database changes.
SQLM_ROLLBACK_ACT	Rollback Active: The unit of work is rolling back its database changes.
SQLM_RECOMP	Recompiling: The database manager is recompiling (that is, rebinding) a plan on behalf of the application.

API Constant	Description
SQLM_COMP	Compiling: The database manager is compiling an SQL statement or precompiling a plan on behalf of the application.
SQLM_INTR	Request Interrupted: An interrupt of a request is in progress.
SQLM_DISCONNECTPEND	Database Disconnect Pending: 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	Decoupled from Agent: 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	Transaction Prepared: The unit of work is part of a global transaction that has entered the prepared phase of the two-phase commit protocol.
SQLM_THCOMT	Transaction Heuristically Committed: The unit of work is part of a global transaction that has been heuristically committed.
SQLM_THABRT	Transaction Heuristically Rolled Back: The unit of work is part of a global transaction that has been heuristically rolled-back.
SQLM_TEND	Transaction Ended: 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	Creating Database: The agent has initiated a request to create a database and that request has not yet completed.
SQLM_RESTART	Restarting Database: The application is restarting a database in order to perform crash recovery.
SQLM_RESTORE	Restoring Database: The application is restoring a backup image to the database.
SQLM_BACKUP	Backing Up Database: The application is performing a backup of the database.
SQLM_LOAD	Data Fast Load: The application is performing a "fast load" of data into the database.
SQLM_UNLOAD	Data Fast Unload: The application is performing a "fast unload" of data from the database.
SQLM_IOERROR_WAIT	Wait to Disable Table space: 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	Quiescing a Table space: The application is performing a quiesce table space request.
SQLM_WAITFOR_REMOTE	Pending remote request: The application is waiting for a response from a remote partition in a partitioned database instance.
SQLM_REMOTE_RQST	Federated request pending: The application is waiting for results from a federated data source.
SQLM_ROLLBACK_TO_SAVEPOINT	Rollback to savepoint: 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[®] 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.

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	-

Description

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.

In an explicit trusted connection, the **auth_id** value does not change immediately when you switch users. Rather, the **auth_id** is updated the

first time you access the database after switching users. This is because the switch user operation is always chained to the subsequent operation.

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 201.

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_SYSMANT

The following defines from sql.h may be used to determine indirect authorizations inherited from group or public:

- SQL_SYSADM_GRP
- SQL_DBADM_GRP
- SQL_CREATETAB_GRP
- SQL_BINDADD_GRP
- SQL_CONNECT_GRP
- SQL_CREATE_EXT_RT_GRP
- SQL_CREATE_NOT_FENC_GRP
- SQL_SYSCTRL_GRP
- SQL_SYSMANT_GRP

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

Table 103. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	memory_pool	Basic
Database Manager	fcm	Basic
Database Manager	fcm_node	Basic
Database Manager	utility_info	Basic
Database	detail_log	Basic
Buffer Pool	bufferpool_nodeinfo	Buffer Pool
Table Space	rollforward	Basic
Lock	lock	Basic
Lock	lock_wait	Basic
Database	db_sto_path_info	Buffer Pool

Table 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

watermark

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	-

Table 146. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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 *sheapthres* configuration parameter. If the element value is greater than or equal to *sheapthres*, it means that the sorts are not getting the full sort heap as defined by the *sortheap* parameter.

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
piped_sorts_requested

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 (*sheaphres*) 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 *piped_sorts_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*.

piped_sorts_accepted - Piped Sorts Accepted

The number of piped sorts that have been accepted.

Element identifier
piped_sorts_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*
- *sheaphres*

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.

total_sorts - Total Sorts

The total number of sorts that have been executed.

Element identifier

total_sorts

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:

```
total_sort_time / 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 soft 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 `sorheap` 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 `sorheap` 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 *fcm_num_channels* configuration parameter to determine the current connection entry utilization. You can use this information to tune *fcm_num_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	fcm	Basic

Usage Use this element in conjunction with the *fcm_num_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

Table 197. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Buffer Pool
Application	stmt	Buffer Pool
Dynamic SQL	dynsql	Buffer Pool, Statement

For snapshot monitoring, this counter can be reset.

Table 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 239.

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_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.

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 239.

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

Table 201. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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 241 and “pool_async_data_reads - Buffer pool asynchronous data reads monitor element” on page 257 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_tablespaces	-
Connection	event_conn	-

Table 204. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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 242.

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
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_tablespace	-
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	-

Table 208. 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

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 241.

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	-

Table 210. 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 242.

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 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 246 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	-
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.

For information about how to use this element, see “pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads” on page 242.

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

Table 215. 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 216. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespaces	-
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_i_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	-
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 239.

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
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 239.

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 250 and “pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads” on page 260.

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 223. 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 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 252.

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	-

Table 226. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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
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:

$$1 - ((\text{pool_data_p_reads} + \text{pool_index_p_reads}) - (\text{pool_async_data_reads} + \text{pool_async_index_reads})) / (\text{pool_data_l_reads} + \text{pool_index_l_reads})$$

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the **num_ioservers** configuration parameter.

Asynchronous reads are performed by database manager prefetchers. 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

Table 237. 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 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_iocleaners` 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:

$$1 - ((\text{pool_data_p_reads} + \text{pool_index_p_reads}) - (\text{pool_async_data_reads} + \text{pool_async_index_reads})) / (\text{pool_data_l_reads} + \text{pool_index_l_reads})$$

By comparing the ratio of asynchronous to synchronous reads, you can gain insight into how well the prefetchers are working. This element can be helpful when you are tuning the **num_ioservers** configuration parameter (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 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_tablespace	-

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_tablespace	-

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_iocleaners* 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 245. 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 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	-
Tablespaces	event_tablespaces	-

Description

The number of read request by the prefetcher to the operating system. These requests are typically large block IOs of multiple pages.

Usage To calculate the average number of data pages in each read request, use the following formula:

$$\text{pool_async_data_reads} / \text{pool_async_data_read_reqs}$$

This average can help you determine the average read IO size used by the prefetcher. This data can also be helpful in understanding the large block IO requirements of the measured workload.

The maximum size of a prefetcher read IO is the extentsize of the table space involved, but it can be smaller under some circumstances:

- when some pages of the extent are already in the bufferpool
- when exceeding operating system capabilities
- when the extentsize is very large, such that doing a large IO would be detrimental to overall performance

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:

$$\frac{\text{pool_drty_pg_steal_clns}}{(\text{pool_drty_pg_steal_clns} + \text{pool_drty_pg_thrsh_clns} + \text{pool_lsln_gap_clns})}$$

If this ratio is low, it may indicate that you have defined too many page cleaners. If your *chnpgs_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_drty_pg_steal_clns* monitor element inserts 0 into the monitor stream.
- There is no explicit triggering of the page cleaners when a synchronous write is needed during victim buffer replacement. To determine whether or not the right number of page cleaners is configured for the database or for specific buffer pools, 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 *chnngpgs_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 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_tablespaces	-
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 I/O

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:

$$\text{direct_reads} / \text{direct_read_reqs}$$

When using system monitors to track I/O, this element helps you distinguish database I/O from non-database I/O on the device.

Direct reads are performed in units, the smallest being a 512-byte sector. They are used when:

- Reading LONG VARCHAR columns
- Reading LOB (large object) columns
- Performing a backup

direct_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.

$$\text{direct_writes} / \text{direct_write_reqs}$$

When using system monitors to track I/O, this element helps you distinguish database I/O from non-database I/O on the device.

Direct writes are performed in units, the smallest being a 512-byte sector. They are used when:

- Writing LONG VARCHAR columns
- Writing LOB (large object) columns
- Performing a restore
- Performing a load.
- Allocating new extents for SMS table space if MPFA is enabled (which is the default)

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	-
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
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
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:

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:

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

Table 304. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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:

$\text{logprimary} \times \text{logfilesiz} \times 4096$ (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:

- logfilesiz
- 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:

- logfilesiz
- 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
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	-
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	-

Table 393. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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_LOCKRELFMAGS_SQLCOMPILER	Locks by SQL compiler.
SQLM_LOCKRELFMAGS_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
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 434. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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
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) 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) 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) 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	<code>tablespace_nodeinfo</code>	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	<code>tablespace_nodeinfo</code>	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	<code>tablespace_nodeinfo</code>	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) 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) 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	dcx_dbase	Basic
DCS Application	dcx_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 History Values	event_stmt_history	-
Deadlocks with Details History Values	event_data_value	-
Deadlocks with Details 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 History Values	event_stmt_history	timestamp
Deadlocks with Details 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 History Values	event_stmt_history	-
Deadlocks with Details 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 History Values	event_stmt_history	-
Deadlocks with Details 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 History Values	event_stmt_history	-
Deadlocks with Details 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 History Values	event_stmt_history	-
Deadlocks with Details 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 History Values	event_stmt_history	-
Deadlocks with Details 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 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 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 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 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 note:

Use this element to determine the health of the HADR connection.

A heartbeat is a message sent from the other HADR database at regular intervals. If the value for this element is zero, no heartbeats have been missed and the connection is healthy. The higher the value, the worse the condition of the connection.

In disconnected mode, heartbeats missed is always shown as 0, because it is not applicable.

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.

dcс_db_name - DCS Database Name

The name of the DCS database as cataloged in the DCS directory.

Element identifier

dcс_db_name

Element type

information

Table 735. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcс_dbase	Basic
DCS Application	dcс_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	dcс_dbase	Basic
DCS Application	dcс_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	dc_s_dbase	Timestamp
DCS Application	dc_s_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	dc_s_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	dc_s_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	dc_s_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	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_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	dcx_dbase	Basic
DCS Application	dcx_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	dc_s_appl	Statement

Usage Use this element to assess how much memory is being allocated. The amount of memory allocated by the DB2 client, DB2 Connect, or the database agent on the target database is related to the number of cursors that are currently open. Knowing this information can help with capacity planning. For example, each open cursor that is doing blocking has a buffer size of RQRIOBLK. If *deferred_prepare* is enabled, then two buffers will be allocated.

This element does not include cursors that were closed by an early close. An early close occurs when the host database returns the last record to the client. The cursor is closed at the host and gateway, but is still open at the client. Early close cursors can be set using the DB2 Call Level Interface.

dc_s_appl_status - DCS Application Status

The status of a DCS application at the DB2 Connect gateway.

Element identifier

dc_s_appl_status

Element type

information

Table 748. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dc_s_appl_info	Basic

Usage Use this element for problem determination on DCS applications. Values are:

- **SQLM_DCS_CONNECTPEND_OUTBOUND**
The application has initiated a database connection from the DB2 Connect gateway to the host database, but the request has not completed yet.
- **SQLM_DCS_UOWWAIT_OUTBOUND**
The DB2 Connect gateway is waiting for the host database to reply to the application's request.
- **SQLM_DCS_UOWWAIT_INBOUND**
The connection from the DB2 Connect gateway to the host database has been established and the gateway is waiting for SQL requests from the application. Or the DB2 Connect gateway is waiting on behalf of the unit of work in the application. This usually means that the application's code is being executed.

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	dcs_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	dcs_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	dcs_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	dcs_appl	Basic
DCS Statement	dcs_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	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
DCS Statement	dcs_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).

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	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_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	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_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	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_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	dcс_dbase	Statement
DCS Application	dcс_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Usage Use this element to get a better idea of the database activity and network traffic at the database or application levels.

max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes

This element represents the number of statements or chains with outbound bytes received between 257 and 512 inclusive.

Element identifier

max_data_received_512

Element type

counter

Table 767. 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_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
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_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	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_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	dcs_dbase	Statement

Table 770. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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_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	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_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	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_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
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

Table 777. 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_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	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_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	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Usage Use this element to get a better idea of the database activity and network traffic at the database or application levels.

max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes

This element represents the number of statements or chains with outbound bytes sent greater than 64000.

Element identifier
max_data_sent_gt64000

Element type
counter

Table 782. 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_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	dcс_dbase	Statement
DCS Application	dcс_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Usage Use this element to get a better idea of the database activity and network traffic at the database or application levels.

max_network_time_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	dcс_dbase	Statement

Table 784. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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	dc_s_dbase	Statement
DCS Application	dc_s_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Usage Use this element to get a better idea of the database activity and network traffic at the database or application levels.

max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms

This element represents the number of statements or chains whose network time was greater than 4 milliseconds but less or equal to 16 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

Element identifier

max_network_time_16_ms

Element type

counter

Table 786. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dc_s_dbase	Statement
DCS Application	dc_s_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Usage Use this element to get a better idea of the database activity and network traffic at the database or application levels.

max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms

This element represents the number of statements or chains whose network time was greater than 16 milliseconds but less or equal to 100 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

Element identifier

max_network_time_100_ms

Element type

counter

Table 787. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcс_dbase	Statement
DCS Application	dcс_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Usage Use this element to get a better idea of the database activity and network traffic at the database or application levels.

max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms

This element represents the number of statements or chains whose network time was greater than 100 milliseconds but less or equal to 500 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

Element identifier

max_network_time_500_ms

Element type

counter

Table 788. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcс_dbase	Statement
DCS Application	dcс_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Usage Use this element to get a better idea of the database activity and network traffic at the database or application levels.

max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms

This element represents the number of statements or chains whose network time was greater than 500 milliseconds. (Network time is the difference between host response time and elapsed execution time for a statement or chain.)

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	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.

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	dcс_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

Table 793. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement, Timestamp
DCS Database	dcс_dbase	Statement, Timestamp
DCS Application	dcс_appl	Statement, Timestamp
DCS Statement	dcс_stmt	Statement, Timestamp
Data Transmission	stmt_transmissions	Statement, Timestamp

For snapshot monitoring at the statement level, this counter cannot be reset. This counter can be reset at other levels.

Usage Use this element, along with other elapsed time monitor elements, to evaluate the database server's processing of SQL requests and to help isolate performance issues.

Subtract this element from the `host_response_time` element to calculate the network elapsed time between DB2 Connect and the host database server.

Note: For the `dcс_dbase`, `dcс_appl`, `dcс_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	dcс_dbase	Statement
DCS Application	dcс_appl	Statement, Timestamp
DCS Statement	dcс_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	dc_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	dc_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	dc_s_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	dc_s_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	-
Transaction	event_xact	-
Deadlock	event_dlconn	-

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	-
Transaction	event_xact	-
Deadlock	event_dlconn	-

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	dcs_appl	Basic
Activities	event_activity	-
Transaction	event_xact	-
Deadlock	event_dlconn	-

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_actng 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	dcs_appl	Basic
Activities	event_activity	-
Transaction	event_xact	-
Deadlock	event_dlconn	-

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
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:

```
write_activity =  
  (INSERT statements + UPDATE statements + DELETE statements ) /  
  (SELECT statements + INSERT statements + UPDATE statements +  
  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:

```
write_activity =  
  (INSERT statements + UPDATE statements + DELETE statements ) /  
  (SELECT statements + INSERT statements + UPDATE statements +  
  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

Table 820. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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	-

Table 828. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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, which includes requests on the coordinator partition, and any subrequests on both coordinator and non-coordinator partitions (like RPC requests or SMP subagent requests). Requests included may or may not be associated with an activity: Both PREPARE and OPEN requests are included in this histogram, for example, but while OPEN requests are always associated with a cursor activity, PREPARE requests are not part of any activity.

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	-

Table 872. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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 in KB for the temporary table space usage of DML activities at all nesting levels in a service class or work class. For service classes, this monitor element returns -1 when COLLECT AGGREGATE ACTIVITY DATA for the service class is set to NONE. For work classes, this monitor elements returns -1 if no COLLECT AGGREGATE ACTIVITY DATA work action is specified for the work class.

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.

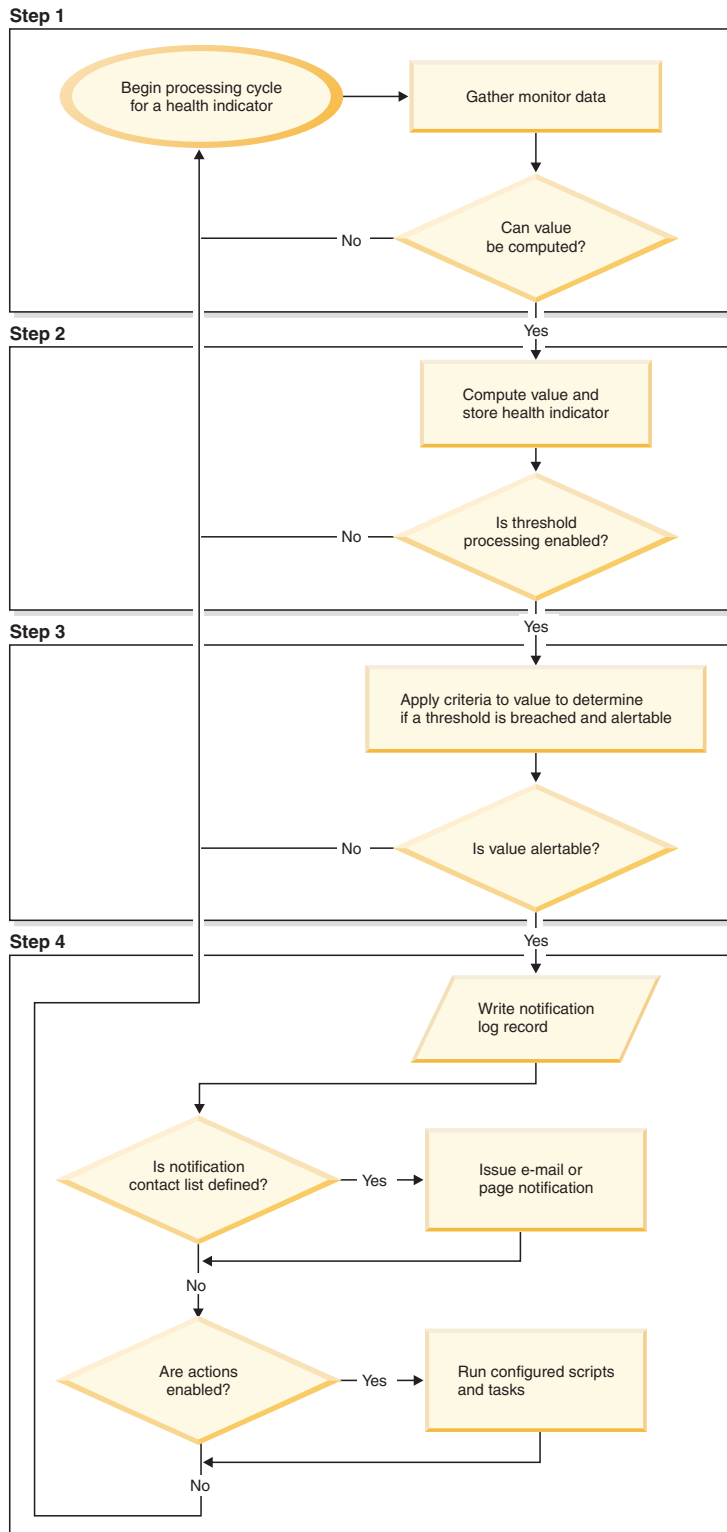


Figure 5. Health indicator process cycle

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.

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
```

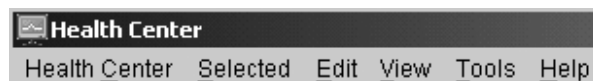
The following list categorizes some of the key tasks that you can do with the Health Center:

- “Enabling health alert notification” on page 518
 - Specifying contact settings and notification configuration parameters
 - Troubleshooting health alert notification
- “Configuring health indicators using the Health Center” on page 547
 - 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 540
 - 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:

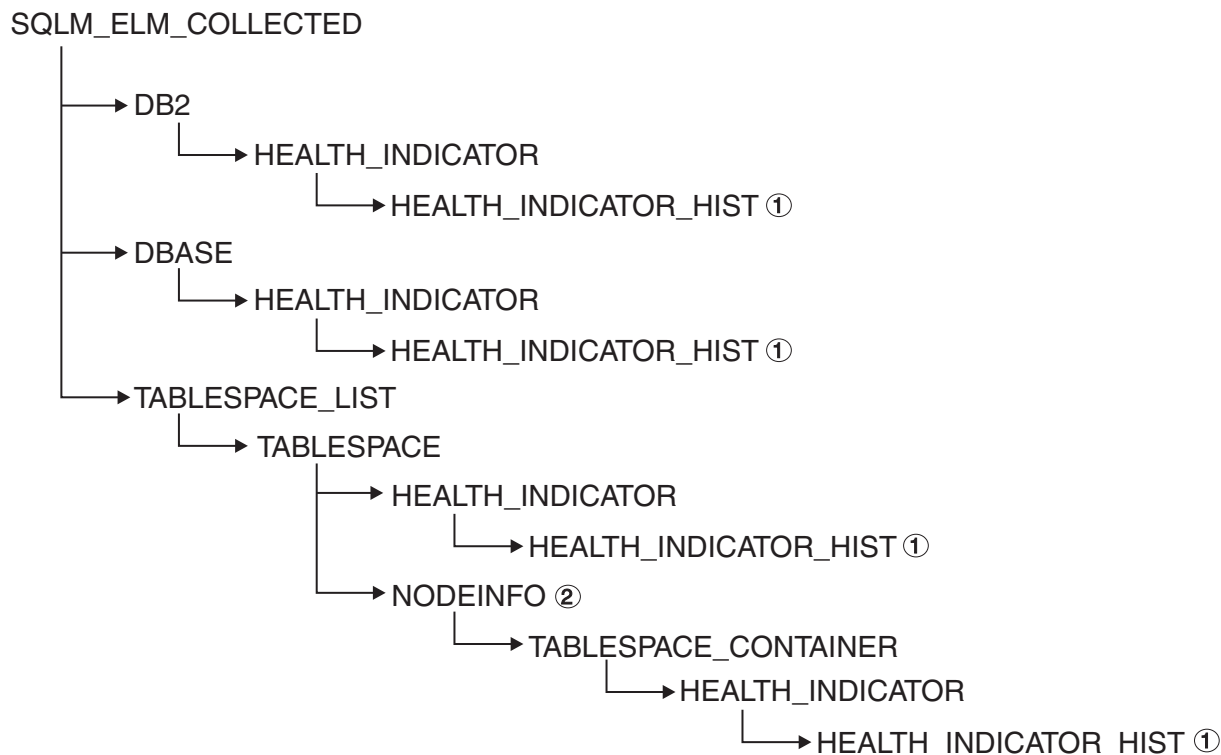


Figure 6. Health snapshot self-describing data stream

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

- 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.

- Before you begin, ensure that a database connection exists, and that the database is quiesced.
- Take the database manager snapshot, using the GET HEALTH SNAPSHOT command:
db2 get health snapshot for db on sample
- 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"> • db2.db2_op_status • db2.sort_privmem_util • db2.mon_heap_util • db.db_op_status • db.sort_shrmem_util • db.spilled_sorts • db.log_util • db.log_fs_util • db.locklist_util • db.apps_waiting_locks • db.db_heap_util • db.db_backup_req • ts.ts_util 	<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"> • db.max_sort_shrmem_util • db.pkgcache_hitratio • db.catcache_hitratio • db.shrworkspace_hitratio 	<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"> • db.deadlock_rate • db.lock_escal_rate 	<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"> • ts.ts_op_status • tsc.tscont_op_status • tsc.tscont_util 	<p>These health indicators is not aggregated.</p>
<ul style="list-style-type: none"> • db.hadr_op_status • db.hadr_log_delay 	<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">• db.tb_reorg_req• db.tb_runstats_req• db.fed_nicknames_op_status• db.fed_servers_op_status	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> and <code>defaultType = DB2ALERTCFG_NOT_DEFAULT</code> or <code>objType = DB2ALERTCFG_OBJTYPE_{DATABASE TABLESPACE CONTAINER}</code> and <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 passwd')
```

In the following example, an alert action defined using a combination of states (QUIESCED:share + QUIESCED:update = 3) is executed if and only if the table space state is both QUIESCED:share and QUIESCED:update.

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
set actionsenabled yes')
```

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
add action script /home/guest001/script1 type operating system command line
parameters userParam working directory /home/guest001/
on attention 3 on aix1 user guest001 using passwd')
```

Starting in DB2 V9.1, health monitor alert actions defined on an object with the same action attributes (name, working directory, command line parameters, host, user and password) run only once, even if it was defined on multiple alert states.

In the following example, the same action is defined on two different alert states. The action is only executed once for a given table space, even if the table space state is in both QUIESCED:share and QUIESCED:update.

```
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
add action script /home/guest001/script1 type operating system command line
parameters userParam working directory /home/guest001/
on attention QUIESCED_SHARE on aix1 user guest001 using passwd')
db2 call SYSPROC.ADMCMD('update alert cfg for tablespaces using ts.ts_op_status
add action script /home/guest001/script1 type operating system command line
parameters userParam working directory /home/guest001/
on attention QUIESCED_UPDATE on aix1 user guest001 using passwd')
```

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

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

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 show detail	HEALTH_DBM_HI_HIS	health_indicator_history
SQLMA_DBASE	get health snapshot for database on <i>dbname</i>	HEALTH_DB_INFO	dbase
		HEALTH_DB_HI	health_indicator
	get health snapshot for database on <i>dbname</i> show detail	HEALTH_DB_HI_HIS	health_indicator_history
SQLMA_DBASE with SQLM_HMON_OPT_COLL_FULL in the agent_id	get health snapshot for database on <i>dbname</i> with full collection	HEALTH_DB_HIC	health_indicator, hi_obj_list
	get health snapshot for database on <i>dbname</i> show detail with full collection	HEALTH_DB_HIC_HIST	health_indicator_history, hi_obj_list
SQLMA_DBASE_ALL	get health snapshot for all databases	HEALTH_DB_INFO	dbase
		HEALTH_DB_HI	health_indicator
	get health snapshot for all databases show detail	HEALTH_DB_HI_HIS	health_indicator_history
SQLMA_DBASE_TABLESPACES	get health snapshot for tablespaces on <i>dbname</i>	HEALTH_TS_INFO	tablespace
		HEALTH_TS_HI	health_indicator
		HEALTH_CONT_INFO	tablespace_container
		HEALTH_CONT_HI	health_indicator
	get health snapshot for tablespaces on <i>dbname</i> show detail	HEALTH_TS_HI_HIS	health_indicator_history
		HEALTH_CONT_HI_HIS	health_indicator_history

The following figure shows the order that logical data groupings can appear in a health snapshot data stream.

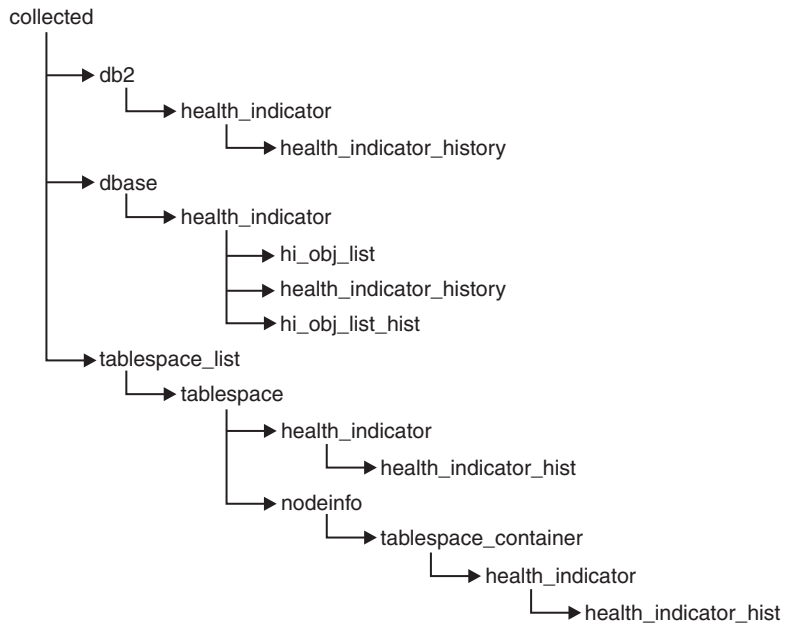


Figure 7. Health snapshot logical data groupings

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 558

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 559
Automatic Resize Table Space Utilization	ts.ts_util_auto_resize	"ts.ts_util_auto_resize - Automatic resize table space utilization health indicator" on page 560
Table Space Utilization	ts.ts_util	"ts.ts_util - Table Space Utilization" on page 560
Table Space Container Utilization	tsc.tscont_util	"tsc.tscont_util - Table Space Container Utilization" on page 561
Table Space Operational State	ts.ts_op_status	"ts.ts_op_status - Table Space Operational State" on page 562
Table Space Container Operational State	tsc.tscont_op_status	"tsc.tscont_op_status - Table Space Container Operational State" on page 562
Table Space Automatic Resize Status	ts.ts_auto_resize_status	"ts.ts_auto_resize_status - Table space automatic resize status health indicator" on page 559

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 562
Shared Sort Memory Utilization	db.sort_shrmem_util	"db.sort_shrmem_util - Shared Sort Memory Utilization" on page 563
Percentage of Sorts That Overflowed	db.spilled_sorts	"db.spilled_sorts - Percentage of Sorts That Overflowed" on page 564
Long Term Shared Sort Memory Utilization	db.max_sort_shrmem_util	"db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization" on page 564

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 565

Table 918. Database manager health indicators (continued)

Name	Identifier	Additional Information
Instance Highest Severity Alert State	–	“Instance Highest Severity Alert State” on page 565

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 566
Database Highest Severity Alert State	–	“Database Highest Severity Alert State” on page 566

Table 920. Maintenance health indicators

Name	Identifier	Additional Information
Reorganization Required	db.tb_reorg_req	“db.tb_reorg_req - Reorganization Required” on page 567
Statistics Collection Required health indicator	db.tb_runstats_req	“db.tb_runstats_req - Statistics Collection Required” on page 567
Database Backup Required	db.db_backup_req	“db.db_backup_req - Database Backup Required” on page 568

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 568
HADR Log Delay health indicator	db.hadr_delay	“db.hadr_delay - HADR Log Delay” on page 569

Table 922. Logging health indicators

Name	Identifier	Additional Information
Log Utilization	db.log_util	“db.log_util - Log Utilization” on page 569
Log File System Utilization	db.log_fs_util	“db.log_fs_util - Log File System Utilization” on page 569

Table 923. Application concurrency health indicators

Name	Identifier	Additional Information
Deadlock Rate	db.deadlock_rate	“db.deadlock_rate - Deadlock Rate” on page 570
Lock List Utilization	db.locklist_util	“db.locklist_util - Lock List Utilization” on page 571
Lock Escalation Rate	db.lock_escal_rate	“db.lock_escal_rate - Lock Escalation Rate” on page 571
Percentage of Applications Waiting on Locks	db.apps_waiting_locks	“db.apps_waiting_locks - Percentage of Applications Waiting on Locks” on page 572

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 573
Package Cache Hit Ratio	db.pkgcache_hitratio	"db.pkgcache_hitratio - Package Cache Hit Ratio" on page 573
Shared Workspace Hit Ratio	db.shrworkspace_hitratio	"db.shrworkspace_hitratio - Shared Workspace Hit Ratio" on page 573

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 574
Database Heap Utilization	db.db_heap_util	"db.db_heap_util - Database Heap Utilization" on page 574

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 575
Data Source Server Status	db.fed_servers_op_status	"db.fed_servers_op_status - Data Source Server Status" on page 575

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). Note: 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 330
- *ts.page_size* is the value of “tablespace_page_size - Table Space Page Size” on page 327
- *ts.max_size* is the value of “tablespace_max_size - Maximum table space size” on page 337

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 330
- *ts.usable* is the value of “tablespace_usable_pages - Usable Pages in Table Space” on page 329

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 soft 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[®].

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks[®] 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. 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.

If you would like to help IBM make the IBM Information Management products easier to use, take the Consumability Survey: <http://www.ibm.com/software/data/info/consumability-survey/>.

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. 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.

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

Name	Form Number	Available in print	Last updated
<i>Administrative API Reference</i>	SC23-5842-02	Yes	April, 2009
<i>Administrative Routines and Views</i>	SC23-5843-02	No	April, 2009
<i>Call Level Interface Guide and Reference, Volume 1</i>	SC23-5844-02	Yes	April, 2009
<i>Call Level Interface Guide and Reference, Volume 2</i>	SC23-5845-02	Yes	April, 2009
<i>Command Reference</i>	SC23-5846-02	Yes	April, 2009
<i>Data Movement Utilities Guide and Reference</i>	SC23-5847-02	Yes	April, 2009
<i>Data Recovery and High Availability Guide and Reference</i>	SC23-5848-02	Yes	April, 2009
<i>Data Servers, Databases, and Database Objects Guide</i>	SC23-5849-02	Yes	April, 2009
<i>Database Security Guide</i>	SC23-5850-02	Yes	April, 2009
<i>Developing ADO.NET and OLE DB Applications</i>	SC23-5851-02	Yes	April, 2009
<i>Developing Embedded SQL Applications</i>	SC23-5852-02	Yes	April, 2009
<i>Developing Java Applications</i>	SC23-5853-02	Yes	April, 2009
<i>Developing Perl and PHP Applications</i>	SC23-5854-02	No	April, 2009
<i>Developing User-defined Routines (SQL and External)</i>	SC23-5855-02	Yes	April, 2009
<i>Getting Started with Database Application Development</i>	GC23-5856-02	Yes	April, 2009
<i>Getting Started with DB2 installation and administration on Linux and Windows</i>	GC23-5857-02	Yes	April, 2009
<i>Internationalization Guide</i>	SC23-5858-02	Yes	April, 2009

Table 934. DB2 technical information (continued)

Name	Form Number	Available in print	Last updated
<i>Message Reference, Volume 1</i>	GI11-7855-01	No	April, 2009
<i>Message Reference, Volume 2</i>	GI11-7856-01	No	April, 2009
<i>Migration Guide</i>	GC23-5859-02	Yes	April, 2009
<i>Net Search Extender Administration and User's Guide</i>	SC23-8509-02	Yes	April, 2009
<i>Partitioning and Clustering Guide</i>	SC23-5860-02	Yes	April, 2009
<i>Query Patroller Administration and User's Guide</i>	SC23-8507-01	Yes	April, 2009
<i>Quick Beginnings for IBM Data Server Clients</i>	GC23-5863-02	No	April, 2009
<i>Quick Beginnings for DB2 Servers</i>	GC23-5864-02	Yes	April, 2009
<i>Spatial Extender and Geodetic Data Management Feature User's Guide and Reference</i>	SC23-8508-02	Yes	April, 2009
<i>SQL Reference, Volume 1</i>	SC23-5861-02	Yes	April, 2009
<i>SQL Reference, Volume 2</i>	SC23-5862-02	Yes	April, 2009
<i>System Monitor Guide and Reference</i>	SC23-5865-02	Yes	April, 2009
<i>Text Search Guide</i>	SC23-5866-01	Yes	April, 2009
<i>Troubleshooting Guide</i>	GI11-7857-02	No	April, 2009
<i>Tuning Database Performance</i>	SC23-5867-02	Yes	April, 2009
<i>Visual Explain Tutorial</i>	SC23-5868-00	No	
<i>What's New</i>	SC23-5869-02	Yes	April, 2009
<i>Workload Manager Guide and Reference</i>	SC23-5870-02	Yes	April, 2009
<i>pureXML Guide</i>	SC23-5871-02	Yes	April, 2009
<i>XQuery Reference</i>	SC23-5872-02	No	April, 2009

Table 935. DB2 Connect-specific technical information

Name	Form Number	Available in print	Last updated
<i>Quick Beginnings for DB2 Connect Personal Edition</i>	GC23-5839-02	Yes	April, 2009
<i>Quick Beginnings for DB2 Connect Servers</i>	GC23-5840-02	Yes	April, 2009
<i>DB2 Connect User's Guide</i>	SC23-5841-02	Yes	April, 2009

Table 936. Information Integration technical information

Name	Form Number	Available in print	Last updated
Information Integration: Administration Guide for Federated Systems	SC19-1020-01	Yes	March, 2008
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-02	Yes	March, 2008
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-01	No	
Information Integration: SQL Replication Guide and Reference	SC19-1030-01	Yes	March, 2008
Information Integration: Introduction to Replication and Event Publishing	SC19-1028-01	Yes	March, 2008

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
 - 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 583.

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 database products.

DB2 documentation

Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Database fundamentals 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 database 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/support/db2_9/

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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks 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.
- Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries. Intel trademark information
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

Special characters

.db2toprc configuration file 18

A

activation time

last_wlm_reset monitor element 489

activities

monitor elements

act_total 477

activity_collected 474

activity_id 475

activity_secondary_id 475

activity_type 476

coord_act_aborted_total 480

coord_act_completed_total 481

coord_act_rejected_total 482

Activity Monitor

overview 81

setup 87

administrative views

APPL_PERFORMANCE scenario 84

BP_HITRATIO scenario 86

BP_READ_IO scenario 86

BP_WRITE_IO scenario 86

LONG_RUNNING_SQL scenario 84

QUERY_PREP_COST scenario 84

TOP_DYNAMIC_SQL

scenario 84

agents

monitor elements

agent_id 184

agent_id_holding_lock 320

agent_pid 208

agent_status 440

agent_sys_cpu_time 404

agent_usr_cpu_time 404

agents_created_empty_pool 217

agents_from_pool 216

agents_registered 214

agents_registered_top 215

agents_stolen 218

agents_top 403

agents_waiting_on_token 214

agents_waiting_top 215

appl_priority 199

associated agent 209

associated_agents_top 218

coord_agent_pid 209

coord_agents_top 217

coordinator agent 209

idle_agents 216

locks_waiting 319

max_agent_overflows 219

num_agents 403

num_assoc_agents 219

primed agent 209

priv_workspace_size_top element 286

quiescer_agent_id 339

rolled_back_agent_id 322

agents (*continued*)

monitor elements (*continued*)

subagent 209

alert actions

health indicators

states 548

alert thresholds

configuration

Health Center 547

alerts

enabling 518

resolving

GET RECOMMENDATIONS command 539

Health Center 540

SQL queries 535

retrieving recommendations

client application 535

aliases

monitor elements

input_db_alias element 409

API request types

health monitor 579

snapshot monitor 39

application creator monitor element 382

applications

monitor elements

appl_id 189

appl_id_holding_lk 321

appl_id_oldest_xact 188

appl_idle_time 208

appl_name 189

appl_priority 199

appl_priority_type 200

appl_section_inserts 289

appl_section_lookups 289

appl_status 185

appls_cur_cons 213

appls_in_db2 214

overview 184

rolled_back_participant_no 313

tpmon_client_app 464

attributes

monitor elements

progress_list_attr monitor element 421

authorization ID

monitor element

session_auth_id element 193

monitor elements

auth_id 192

execution_id element 196

quiescer_auth_id 339

authorizations

user authorization level monitor element

authority_lvl element 200

automatic storage path

monitor elements

db_storage_path 181

sto_path_free_sz 182

B

- backup
 - health indicators
 - db.db_backup_req 568
 - monitor elements
 - last_backup 181
 - requirements
 - health indicator 568
- books
 - printed
 - ordering 586
- buffer pools
 - monitor elements
 - activity 239
 - block_ios 270
 - bp_cur_buffsz 271
 - bp_id 241
 - bp_name 268
 - bp_new_buffsz 272
 - bp_pages_left_to_remove 272
 - bp_tbsp_use_count 272
 - buff_free 237
 - buff_free_bottom 237
 - pool_async_data_read_reqs 263
 - pool_async_data_reads 257
 - pool_async_data_writes 258
 - pool_async_index_read_reqs 264
 - pool_async_index_reads 259
 - pool_async_index_writes 258
 - pool_async_read_time 261
 - pool_async_write_time 262
 - pool_async_xda_read_reqs 264
 - pool_async_xda_reads 260
 - pool_async_xda_writes 261
 - pool_data_l_reads 241
 - pool_data_p_reads 243
 - pool_data_writes 245
 - pool_drty_pg_steal_clns 265
 - pool_drty_pg_thrsh_clns 267
 - pool_index_l_reads 246
 - pool_index_p_reads 248
 - pool_index_writes 249
 - pool_lsn_gap_clns 265
 - pool_no_victim_buffer 266
 - pool_read_time 255
 - pool_temp_data_l_reads 242
 - pool_temp_data_p_reads 244
 - pool_temp_index_l_reads 247
 - pool_temp_index_p_reads 249
 - pool_temp_xda_l_reads 252
 - pool_temp_xda_p_reads 253
 - pool_write_time 256
 - pool_xda_l_reads 251
 - pool_xda_p_reads 253
 - pool_xda_writes 254
 - tablespace_cur_pool_id 328
 - tablespace_next_pool_id 329
 - monitoring
 - administrative views 86
- BUFFERPOOLS event type
 - overview 49
- buffers
 - monitor elements
 - num_log_data_found_in_buffer 299
- byte order
 - monitor elements
 - byte_order 411

C

- caching
 - stats_cache_size monitor element 506
- capturing health snapshots
 - using a client application 528
 - using CLP 527
 - using SQL 526
- catalog cache
 - health indicators
 - db.catcache_hitratio 573
 - monitor elements
 - cat_cache_inserts 278
 - cat_cache_lookups 277
 - cat_cache_overflows 278
 - cat_cache_size_top 279
 - overview 276
- catalog nodes
 - monitor elements
 - catalog_node 180
 - catalog_node_name 179
- CCSID (coded character set identifier)
 - monitor elements
 - host_ccsid 440
- channels currently free monitor element
 - FCM (Fast Communications Manager)
 - ch_free monitor element 239
- chargeback 68
- client applications
 - capturing health snapshots 528
- Client identification 68
- client operating platform monitor element
 - client_platform element 198
- client process ID monitor element
 - client_pid element 197
- client product and version ID monitor element
 - client_prdid element 193
- CLP (command line processor)
 - capturing health snapshots 527
 - commands
 - health monitor 578
- code pages
 - monitor elements
 - codepage_id 187
 - host_ccsid 440
- commit
 - monitor elements
 - int_commits element 373
- commit statements attempted monitor element
 - commit_sql_stmts element 369
- communication error time monitor element
 - gw_comm_error_time element 462
- communication errors monitor element
 - gw_comm_errors element 461
- communication protocol
 - monitor elements
 - client_protocol 198
- completed progress work units monitor element
 - monitor elements
 - progress_completed_units element 421
- con_response_time monitor element 461
- configuring
 - db2toprc 18
- connections
 - monitor elements
 - appl_con_time 204
 - appls_cur_cons 213
 - appls_in_db2 214

- connections (*continued*)
 - monitor elements (*continued*)
 - con_elapsed_time 461
 - con_local_databases 212
 - conn_complete_time 204
 - conn_time 178
 - connection_status 238
 - connections_top 204
 - dl_conns 310
 - gw_connections_top 435
 - gw_cons_wait_client 437
 - gw_cons_wait_host 436
 - gw_cur_cons 436
 - gw_total_cons 435
 - local_cons 211
 - local_cons_in_exec 212
 - num_gw_conn_switches 220
 - overview 209
 - rem_cons_in 210
 - rem_cons_in_exec 211
 - total_sec_cons 219
- CONNECTIONS event type
 - overview 49
- containers
 - monitor elements
 - container_accessible monitor element 343
 - container_id monitor element 341
 - container_name monitor element 341
 - container_total_pages monitor element 342
 - container_type monitor element 341
 - container_usable_pages monitor element 342
- counters
 - data element type 9
- CPU time
 - monitor elements
 - agent_sys_cpu_time 404
 - agent_usr_cpu_time 404
 - ss_sys_cpu_time 407
 - ss_usr_cpu_time 407
 - stmt_sys_cpu_time 405
 - stmt_usr_cpu_time 405
 - system_cpu_time 406
 - tot_s_cpu_time 408
 - tot_u_cpu_time 408
 - user_cpu_time 406
- CREATE EVENT MONITOR statement
 - event types 49
- creator monitor element 382
- cursors
 - monitor elements
 - acc_curs_blk 366
 - blocking_cursor 462
 - cursor_name 381
 - open_cursors 439
 - open_loc_curs 366
 - open_loc_curs_blk 366
 - open_rem_curs 364
 - open_rem_curs_blk 365
 - rej_curs_blk 365
 - SQL 364
- custom controls
 - accessing 521

D

- data
 - element types
 - counters 9
 - overview 8
 - fragmentation
 - overflow_accesses monitor element 353
 - data partitions
 - data partition identifier monitor element 302
 - data sources
 - data source name monitor element 465
 - health indicator 575
 - data_partition_id monitor element 302
 - database configuration monitor elements
 - overview 239
 - DATABASE event type 49
 - database manager monitor elements
 - overview 209
 - server_db2_type element 173
 - database paths
 - db_path element monitor element 177
 - database system events
 - information collection 51
 - database system monitor
 - data organization 8
 - information
 - restricting 21
 - interfaces 511
 - memory requirements 11
 - output 10
 - overview 3
 - sample 511
 - self-describing data stream 10
 - database-managed space (DMS)
 - table spaces
 - health indicators 558
 - databases
 - aliases
 - application monitor element 194
 - gateway monitor element 434
 - connections
 - connects since database activation monitor element 213
 - monitor elements
 - application 194
 - connects since database activation 213
 - database deactivation timestamp 178
 - database files closed 256
 - database operated on by utility 416
 - gateway 434
 - datasource_name element 465
 - db_heap_top monitor element
 - description 290
 - db.lock_escal_rate health indicator 571
 - db.locklist_utilization health indicator 571
 - DB2 Connect
 - monitor elements
 - gw_con_time 435
 - gw_cur_cons 436
 - gw_exec_time 437
 - gw_total_cons 435
 - overview 434
 - DB2 Information Center
 - languages 587
 - updating 588
 - versions 587
 - viewing in different languages 587

- DB2 Performance Counters 105
- db2event.ctl control file 62
- db2perf command
 - resetting database performance values 108
- db2perfi command
 - installing and registering DB2Perf.DLL 105
- db2perfr command
 - registering administrator username and password with DB2 106
- db2top
 - configuration file 18
- db2top monitoring utility 16
- deadlock event monitor 68
- deadlocks
 - db.deadlock_rate health indicator 570
 - event types 49
 - monitor elements
 - deadlock_id 311
 - deadlock_node 312
 - deadlocks 303
 - dl_conns 310
 - int_deadlock_rollbacks 375
 - participant_no 312
- DELETE statement
 - delete_sql_stmts monitor element 467
- descriptors
 - progress_description monitor element 419
- disconn_time element 178
- documentation
 - overview 583
 - PDF 583
 - printed 583
 - terms and conditions of use 590
- dynamic buffer pool monitor elements
 - overview 271
- dynamic SQL
 - monitor elements 401

E

- environment handles
 - comp_env_desc monitor element 393
- errors
 - gw_comm_errors monitor element 461
- event monitors
 - blocked 63
 - buffers 63
 - creating
 - file 60
 - overview 53
 - partitioned database 66
 - pipe 64
 - table 53
 - data transferring between systems 78
 - database system events 51
 - DEADLOCK WITH DETAILS HISTORY 95
 - elements
 - count 410
 - event_monitor_name 412
 - evmon_activates 414
 - evmon_flushes 414
 - overview 410
 - event type to logical data group mappings 145
 - file management 62
 - named pipe management 65
 - non-blocked 63

- event monitors (*continued*)
 - output
 - sample 68
 - self-describing data stream 76
 - overview 49
 - records 75
 - table management 56
- event records
 - finding corresponding applications 75
- events
 - monitor elements
 - event_time 413
 - start_time 383
 - stop_time 383

F

- FCM (Fast Communications Manager)
 - monitor elements
 - buff_free 237
 - buff_free_bottom 237
 - ch_free 239
 - ch_free_bottom 239
 - overview 237
 - total_buffers_rcvd 238
 - total_buffers_sent 238
- federated databases
 - monitor elements 464
- federated servers
 - monitor elements
 - disconnects element 465
- fetching
 - monitor elements
 - fetch_count 386
- file event monitors
 - buffering 63
 - creating 60
 - file management 62
 - formatting output from command line 75
- file systems
 - health indicators
 - db.log_fs_util 569
 - monitor elements
 - fs_caching 335
 - fs_id 183
 - fs_total_size 182
 - fs_type 183
 - fs_used_size 182
- files_closed element 256
- FLUSH EVENT MONITOR statement
 - event types 49
- format
 - health indicator 557

G

- GET SNAPSHOT command
 - sample output 41, 87
- global health snapshots 532
- global snapshots on partitioned database systems 44
- graphical tools
 - health monitor 533
- gw_db_alias element 434

H

- hash joins
 - monitor elements
 - active_hash_joins 233
 - hash_join_overflows 233
 - hash_join_small_overflows 234
 - overview 231
 - post_shrthreshold_hash_joins 232
 - post_threshold_hash_joins 232
 - total_hash_joins 231
- health alerts
 - enabling 518
 - recommendations
 - retrieving using CLP 535
 - resolving
 - client applications 539
 - SQL queries 535
- Health Center
 - health indicators 515, 551
 - interface 521
 - overview 521, 533
 - status beacon 533
 - tasks 521
- health indicators
 - alert actions
 - combined states 548
 - alerts
 - resolving using SQL 535
 - resolving using the Health Center 540
 - retrieving recommendations 535, 539
 - applications waiting on locks 572
 - catalog cache hit ratio 573
 - collection state-based 515, 551
 - configuration
 - client applications 545
 - Health Center 547
 - overview 541
 - resetting 544
 - retrieving 543
 - updates 543
 - data 526
 - databases
 - heap utilization 574
 - highest severity alert state 566
 - operational state 566
 - db.alert_state 566
 - db.apps_waiting_locks 572
 - db.catcache_hitratio 573
 - db.database_heap_util 574
 - db.db_auto_storage_util 558
 - db.db_backup_req 568
 - db.db_op_status 566
 - db.deadlock_rate 570
 - db.fed_nicknames_op_status 575
 - db.fed_servers_op_status 575
 - db.hadr_delay 569
 - db.hadr_op_status 568
 - db.lock_escal_rate 571
 - db.locklist_utilization 571
 - db.log_fs_util 569
 - db.log_util 569
 - db.max_sort_shrmem_util 564
 - db.pkgcache_hitratio 573
 - db.shrworkspace_hitratio 573
 - db.sort_shrmem_util 563
 - db.spilled_sorts 564
 - db.tb_reorg_req 567
- health indicators (*continued*)
 - db.tb_runstats_req 567
 - db2.db2_alert_state 565
 - db2.db2_op_status 565
 - db2.mon_heap_util 574
 - db2.sort_privmem_util 562
 - deadlock rate 570
 - DMS table spaces 558
 - format 557
 - instances
 - highest severity alert state 565
 - operational state 565
 - lock escalation rate 571
 - lock list utilization 571
 - logs
 - file system utilization 569
 - space utilization 569
 - monitor heap utilization 574
 - overview 515, 551
 - package cache hit ratio 573
 - process cycle 517
 - shared workspace hit ratio 573
 - sort memory utilization
 - long-term shared 564
 - private 562
 - shared 563
 - sorts that overflowed 564
 - state-based 515, 551
 - summary 555
 - table spaces
 - container operational state 562
 - container utilization 561
 - operational state 562
 - storage utilization 560
 - threshold-based 515, 551
 - ts.ts_auto_resize_status 559
 - ts.ts_op_status 562
 - ts.ts_util 560
 - ts.ts_util_auto_resize 560
 - tsc.tscont_op_status 562
 - tsc.utilization 561
- health monitor
 - alerts 541
 - API request types 579
 - CLP commands 578
 - description 515
 - graphical tools 533
 - Health Center 533
 - Health Center Status Beacon 533
 - interfaces 577
 - logical data groups 553
 - recommendation retrieval
 - using client application 539
 - using CLP 535
 - using SQL 535
 - sample output 530
 - SQL table functions 578
 - starting 525
 - stopping 525
 - thresholds 541
- health snapshots
 - capturing
 - using client applications 528
 - using CLP 527
 - using SQL table functions 526
 - global 532

- help
 - configuring language 587
 - SQL statements 587
- High Availability Disaster Recovery (HADR)
 - health indicators
 - db.hadr_delay 569
 - db.hadr_op_status 568
 - monitor elements
 - hadr_connect_status 424
 - hadr_connect_time 425
 - hadr_heartbeat 426
 - hadr_local_host 427
 - hadr_local_service 427
 - hadr_log_gap 432
 - hadr_peer_window 433
 - hadr_peer_window_end 433
 - hadr_primary_log_file 430
 - hadr_primary_log_lsn 430
 - hadr_primary_log_page 430
 - hadr_remote_host 428
 - hadr_remote_instance 429
 - hadr_remote_service 428
 - hadr_role 422
 - hadr_standby_log_file 431
 - hadr_standby_log_lsn 432
 - hadr_standby_log_page 431
 - hadr_state 423
 - hadr_syncmode 424
 - hadr_timeout 429
- histograms
 - monitor elements
 - histogram_type 488
 - number_in_bin 490
 - top 502
- host databases
 - host_db_name monitor element 434
 - name monitor element 434

I

- I/O
 - monitor elements
 - num_log_part_page_io 298
 - num_log_write_io 297
 - num_pages_from_block_IOs element 271
 - num_pages_from_vectorized_IOs element 270
 - vectorized_ios 270
- identifiers
 - monitor elements
 - arm_correlator 477
 - bin_id 478
 - db_work_action_set_id 487
 - db_work_class_id 487
 - host_prdid element 194
 - parent_activity_id 490
 - parent_uow_id 490
 - sc_work_action_set_id 494
 - sc_work_class_id 494
 - service_class_id 495
 - sql_req_id element 415
 - work_action_set_id 503
 - work_class_id 504
- indexes
 - index object pages monitor element 357
 - monitor elements
 - index_object_pages element 357
 - reorg_index_id monitor element 363

- Indoubt Transaction Manager
 - overview 13
- indoubt transactions
 - monitoring 13
- insert_timestamp monitor element 384
- inserting data
 - monitor elements
 - appl_section_inserts 289
- instances
 - operational states
 - health indicator 565
- int_rows_deleted monitor element 353

L

- LOBs (large objects)
 - lob_object_pages element 357
- local databases
 - monitor elements
 - con_local_dbases 212
- location
 - monitor elements
 - db_location element 180
- lock list utilization health indicator 571
- lock modes
 - monitor elements
 - lock_current_mode monitor element 316
 - lock_mode element 306
 - lock_mode_requested element 311
- lock waits
 - monitor elements
 - lock_wait_start_time 319
 - lock_wait_time 318
 - lock_waits 317
- lock_escalation element 310
- locks
 - escalation
 - escalation monitor element 310
 - health indicator 571
 - monitor elements
 - agent_id_holding_lock 320
 - appl_id_holding_lk 321
 - lock_attributes 314
 - lock_count 315
 - lock_escals 304
 - lock_hold_count 316
 - lock_list_in_use 302
 - lock_name monitor element 314
 - lock_node element 309
 - lock_object_name element 308
 - lock_object_type element 307
 - lock_release_flags monitor element 315
 - lock_status element 307
 - lock_timeout_val element 320
 - lock_timeouts 309
 - locks_held 301
 - locks_held_top 310
 - locks_in_list 313
 - locks_waiting 319
 - participant_no_holding_lk 313
 - remote_lock_time 473
 - remote_locks 469
 - sequence_no_holding_lk 321
 - stmt_lock_timeout 390
 - uow_lock_wait_time 319
 - x_lock_escals 305

- log buffer
 - monitor elements 298
- log files
 - health indicators
 - db.log_fs_util 569
 - monitor elements
 - current_active_log 300
 - current_archive_log 300
 - first_active_log 299
 - hadr_log_gap 432
 - hadr_primary_log_file 430
 - hadr_primary_log_page 430
 - hadr_standby_log_file 431
 - hadr_standby_log_page 431
 - last_active_log 299
 - log_read_time 296
 - log_reads 292
 - sec_logs_allocated element 292
- log sequence number (LSN)
 - monitor elements
 - hadr_primary_log_lsn 430
 - hadr_standby_log_lsn 432
- log spaces
 - health indicators
 - db.log_util 569
 - monitor elements
 - log_held_by_dirty_pages 295
 - log_to_redo_for_recovery 296
 - log_write_time 296
 - log_writes 293
 - sec_log_used_top element 290
 - smallest_log_avail_node 188
 - tot_log_used_top element 291
 - total_log_available element 294
 - total_log_used element 294
 - uow_log_space_used 293
- logical data groups
 - COLLECT ACTIVITY DATA settings effects 169
 - data organization 8
 - event monitors 148
 - health monitor 553
 - mapping to event types 145
 - snapshot monitor 111
- long data
 - monitor elements
 - long_object_pages element 358

M

- memory
 - health indicators
 - db.sort_shrmem_util 563
 - db2.sort_privmem_util 562
 - monitor elements
 - comm_private_mem 218
 - db_heap_top 290
 - lock_list_in_use 302
 - pool_cur_size 222
 - pool_id 220
 - pool_max_size 223
 - pool_secondary_id 221
 - pool_watermark 223
 - pool
 - monitor elements 220
- memory requirements
 - database system monitor 11
- Memory Visualizer
 - overview 99
 - using 97
- messages
 - monitor elements
 - message 415
 - message_time 416
- minimum channels free monitor element
 - FCM (Fast Communications Manager)
 - ch_free_bottom monitor element 239
- mon_heap_sz database manager configuration parameter 11
- monitor element
 - authorization ID
 - session_auth_id element 193
- monitor elements 299
 - acc_curs_blk 366
 - accessing
 - overflow_accesses element 353
 - activation time
 - last_wlm_reset 489
 - active_sorts 229
 - activities
 - act_total 477
 - activity_collected 474
 - activity_id 475
 - activity_secondary_id 475
 - activity_type 476
 - coord_act_aborted_total 480
 - coord_act_completed_total 481
 - coord_act_rejected_total 482
 - agents
 - agent_id 184
 - agent_id_holding_lock 320
 - agent_pid 208
 - agent_status 440
 - agent_sys_cpu_time 404
 - agent_usr_cpu_time 404
 - agents_created_empty_pool 217
 - agents_from_pool 216
 - agents_registered 214
 - agents_registered_top 215
 - agents_stolen 218
 - agents_top 403
 - agents_waiting_on_token 214
 - agents_waiting_top 215
 - appl_priority 199
 - associated_agents_top 218
 - coord_agent_pid 209
 - coord_agents_top 217
 - idle_agents 216
 - information 208
 - max_agent_overflows 219
 - num_agents 403
 - num_assoc_agents 219
 - overview 209
 - priv_workspace_size_top element 286
 - quiescer_agent_id 339
 - rolled_back_agent_id 322
 - aliases
 - client_db_alias 194
 - input_db_alias element 409
 - applications
 - appl_id 189
 - appl_id_holding_lk 321
 - appl_id_oldest_xact 188
 - appl_idle_time 208
 - appl_name 189

monitor elements (*continued*)

- applications (*continued*)
 - appl_priority_type 200
 - appl_section_inserts 289
 - appl_section_lookups 289
 - appl_status 185
 - overview 184
 - tpmon_client_app 464
- attributes
 - progress_list_attr monitor element 421
- auth_id 192
- authority_bitmap 201
- authorization ID
 - execution_id element 196
- automatic storage path
 - sto_path_free_sz 182
- binds_precompiles 376
- blocking_cursor 462
- blocks_pending_cleanup 301
- buffer pools
 - activity 239
 - block_ios 270
 - bp_cur_buffsz 271
 - bp_id 241
 - bp_name 268
 - bp_new_buffsz 272
 - bp_pages_left_to_remove 272
 - bp_tbsp_use_count 272
 - buff_free 237
 - buff_free_bottom 237
 - pool_async_data_read_reqs 263
 - pool_async_data_reads 257
 - pool_async_data_writes 258
 - pool_async_index_read_reqs 264
 - pool_async_index_reads 259
 - pool_async_index_writes 258
 - pool_async_read_time 261
 - pool_async_write_time 262
 - pool_async_xda_read_reqs 264
 - pool_async_xda_reads 260
 - pool_async_xda_writes 261
 - pool_data_l_reads 241
 - pool_data_p_reads 243
 - pool_data_writes 245
 - pool_drty_pg_steal_clns 265
 - pool_drty_pg_thrsh_clns 267
 - pool_index_l_reads 246
 - pool_index_p_reads 248
 - pool_index_writes 249
 - pool_lsn_gap_clns 265
 - pool_no_victim_buffer 266
 - pool_read_time 255
 - pool_temp_data_l_reads 242
 - pool_temp_data_p_reads 244
 - pool_temp_index_l_reads 247
 - pool_temp_index_p_reads 249
 - pool_temp_xda_l_reads 252
 - pool_temp_xda_p_reads 253
 - pool_write_time 256
 - pool_xda_l_reads 251
 - pool_xda_p_reads 253
 - pool_xda_writes 254
- buffers
 - num_log_data_found_in_buffer 299
- byte order
 - byte_order 411

monitor elements (*continued*)

- caching
 - stats_cache_size 506
- cat_cache_inserts 278
- cat_cache_lookups 277
- cat_cache_overflows 278
- cat_cache_size_top 279
- catalog cache 276
- catalog_node 180
- catalog_node_name 179
- client_pid element 197
- client_platform element 198
- client_prdid element 193
- code pages
 - codepage_id 187
 - host_ccsid 440
- comm_private_mem 218
- commit
 - int_commits element 373
- commit_sql_stmts element 369
- communication protocol
 - client_protocol 198
- completed progress work units monitor element
 - progress_completed_units element 421
- connections
 - appl_con_time 204
 - appls_cur_cons 213
 - appls_in_db2 214
 - con_elapsed_time 461
 - con_local_databases 212
 - conn_complete_time 204
 - conn_time 178
 - connection_status 238
 - connections_top 204
 - gw_connections_top 435
 - gw_cons_wait_client 437
 - gw_cons_wait_host 436
 - gw_cur_cons 436
 - gw_total_cons 435
 - local_cons 211
 - local_cons_in_exec 212
 - num_gw_conn_switches 220
 - overview 209
 - rem_cons_in 210
 - rem_cons_in_exec 211
 - total_sec_cons 219
- container status 341
- containers
 - container_accessible monitor element 343
 - container_id monitor element 341
 - container_name monitor element 341
 - container_total_pages monitor element 342
 - container_type monitor element 341
 - container_usable_pages monitor element 342
- coord_act_est_cost_avg 486
- coord_act_exec_time_avg 484
- coord_act_interarrival_time_avg 486
- coord_act_lifetime_avg 483
- coord_act_queue_time_avg 484
- country_code
 - replaced with territory_code 199
- CPU time
 - ss_sys_cpu_time 407
 - ss_usr_cpu_time 407
 - stmt_sys_cpu_time 405
 - stmt_usr_cpu_time 405
 - system_cpu_time 406

- monitor elements *(continued)*
 - CPU time *(continued)*
 - tot_s_cpu_time 408
 - tot_u_cpu_time 408
 - user_cpu_time 406
 - CPU usage 404
 - cursor_name 381
 - data organization 8
 - data_partition_id 302
 - database and application activity 301
 - database configuration 239
 - database connections
 - total_cons element 213
 - database heap 290
 - database identification and status 176
 - database manager
 - server_db2_type element 173
 - database manager configuration 209
 - database paths
 - db_path element 177
 - database system 171
 - db_heap_top 290
 - db_storage_path 181
 - DB2 Connect
 - gw_con_time 435
 - gw_exec_time 437
 - overview 434
 - deadlocks
 - deadlock_id 311
 - deadlock_node 312
 - deadlocks 303
 - dl_conns 310
 - int_deadlock_rollbacks 375
 - overview 301
 - DELETE statement
 - delete_sql_stmts element 467
 - deleting
 - int_rows_deleted element 353
 - descriptors
 - progress_description element 419
 - Dynamic buffer pool
 - overview 271
 - dynamic SQL 401
 - environment handles
 - comp_env_desc element 393
 - errors
 - gw_comm_errors element 461
 - event monitors
 - count 410
 - event_monitor_name 412
 - evmon_activates 414
 - evmon_flushes 414
 - list 148
 - overview 410
 - events
 - event_time element 413
 - start_time element 383
 - stop_time element 383
 - executing
 - act_exec_time 477
 - fabrications
 - stats_fabricate_time 508
 - stats_fabrications 506
 - fast communications manager 237
 - FCM (Fast Communications Manager)
 - ch_free monitor element 239
 - ch_free_bottom monitor element 239
- monitor elements *(continued)*
 - FCM (Fast Communications Manager) *(continued)*
 - total_buffers_rcvd element 238
 - total_buffers_sent element 238
 - federated database systems 464
 - federated servers
 - disconnects element 465
 - fetching
 - fetch_count 386
 - file systems
 - fs_caching 335
 - fs_id 183
 - fs_total_size 182
 - fs_type 183
 - fs_used_size 182
 - gw_comm_error_time element 462
 - HADR
 - HADR peer window 433
 - HADR peer window end 433
 - HADR (high availability disaster recovery)
 - hadr_connect_status 424
 - hadr_connect_time 425
 - hadr_heartbeat 426
 - hadr_local_host 427
 - hadr_local_service 427
 - hadr_log_gap 432
 - hadr_primary_log_file 430
 - hadr_primary_log_lsn 430
 - hadr_primary_log_page 430
 - hadr_remote_host 428
 - hadr_remote_instance 429
 - hadr_remote_service 428
 - hadr_role 422
 - hadr_standby_log_file 431
 - hadr_standby_log_lsn 432
 - hadr_standby_log_page 431
 - hadr_state 423
 - hadr_syncmode 424
 - hadr_timeout 429
 - overview 422
 - hash joins
 - active_hash_joins 233
 - hash_join_overflows 233
 - hash_join_small_overflows 234
 - overview 231
 - post_shrthreshold_hash_joins 232
 - post_threshold_hash_joins 232
 - total_hash_joins 231
 - histograms
 - histogram_type 488
 - number_in_bin 490
 - top 502
 - host databases
 - host_db_name element 434
 - I/O
 - num_log_part_page_io 298
 - num_log_write_io 297
 - num_pages_from_block_IOs element 271
 - num_pages_from_vectorized_IOs element 270
 - vectorized_ios 270
 - identifiers
 - arm_correlator 477
 - bin_id 478
 - db_work_action_set_id 487
 - db_work_class_id 487
 - host_prdid element 194
 - parent_activity_id 490

monitor elements (*continued*)

- identifiers (*continued*)
 - parent_uow_id 490
 - sc_work_action_set_id 494
 - sc_work_class_id 494
 - service_class_id 495
 - sql_req_id element 415
 - work_action_set_id 503
 - work_class_id 504
- inbound_bytes_received element 442
- inbound_bytes_sent element 443
- inbound_comm_address element 441
- indexes
 - index_object_pages element 357
- insert_timestamp 384
- intra-query parallelism 403
- is_system_appl 195
- LOBs (large objects)
 - lob_object_pages element 357
- location
 - db_location element 180
- lock modes
 - lock_current_mode monitor element 316
 - lock_mode element 306
 - lock_mode_requested element 311
- lock waits
 - lock_wait_time 318
 - lock_waits 317
 - overview 317
- locks
 - lock_attributes 314
 - lock_count 315
 - lock_escals 304
 - lock_hold_count 316
 - lock_list_in_use 302
 - lock_name monitor element 314
 - lock_node element 309
 - lock_object_name element 308
 - lock_object_type element 307
 - lock_release_flags monitor element 315
 - lock_status element 307
 - lock_timeout_val element 320
 - lock_timeouts 309
 - locks_held 301
 - locks_held_top 310
 - locks_in_list 313
 - locks_waiting 319
 - overview 301
 - participant_no_holding_lk 313
 - remote_lock_time 473
 - remote_locks 469
 - sequence_no_holding_lk 321
 - stmt_lock_timeout 390
 - uow_lock_wait_time 319
 - x_lock_escals 305
- log buffers
 - num_log_buffer_full 298
- log files
 - current_active_log 300
 - current_archive_log 300
 - first_active_log 299
 - last_active_log 299
 - log_read_time 296
 - log_reads 292
 - sec_logs_allocated element 292
- log spaces
 - log_held_by_dirty_pages 295

monitor elements (*continued*)

- log spaces (*continued*)
 - log_to_redo_for_recovery 296
 - log_write_time 296
 - log_writes 293
 - sec_log_used_top element 290
 - smallest_log_avail_node 188
 - tot_log_used_top element 291
 - total_log_available element 294
 - total_log_used element 294
 - uow_log_space_used 293
- logging 290
- long data
 - long_object_pages element 358
- memory pool 220
- messages
 - message 415
- names
 - db_name element 176
 - dcs_db_name element 434
 - service_subclass_name 495
 - service_superclass_name 496
 - work_action_set_name 503
 - work_class_name 504
- network time
 - max_network_time_1_ms 454
 - max_network_time_100_ms 456
 - max_network_time_16_ms 455
 - max_network_time_4_ms 455
 - max_network_time_500_ms 456
 - max_network_time_gt500_ms 457
- network_time_bottom 458
- network_time_top 457
- nicknames
 - create_nickname element 467
 - create_nickname_time element 472
- nodes
 - coord_node monitor element 203
 - node_number element 202
 - num_nodes_in_db2_instance 410
 - ss_node_number element 396
- non-buffered I/O activity 272
- num_db_storage_paths 181
- num_indoubt_trans 316
- num_log_part_page_io 298
- num_log_read_io 297
- num_log_write_io 297
- num_nodes_in_db2_instance 410
- num_transmissions 460
- num_transmissions_group 460
- numbers
 - progress_list_cur_seq_num element 419
 - ss_number element 396
- OLAP functions 235, 236, 237
- On-Line Analytical Processing (OLAP)
 - overview 234
- open_cursors 439
- open_loc_curs 366
- open_loc_curs_blk 366
- open_rem_curs 364
- open_rem_curs_blk 365
- operations
 - direct_read_reqs element 274
 - direct_read_time element 275
 - direct_reads element 273
 - direct_write_reqs element 275
 - direct_write_time element 276

monitor elements (*continued*)

- operations (*continued*)
 - direct_writes element 273
 - stmt_operation element 378
- outbound bytes
 - max_data_sent_1024 448
 - max_data_sent_128 445
 - max_data_sent_16384 451
 - max_data_sent_2048 448
 - max_data_sent_256 446
 - max_data_sent_31999 452
 - max_data_sent_4096 449
 - max_data_sent_512 447
 - max_data_sent_64000 453
 - max_data_sent_8192 450
 - max_data_sent_gt64000 454
- outbound bytes received
 - max_data_received_1024 448
 - max_data_received_128 445
 - max_data_received_16384 451
 - max_data_received_2048 449
 - max_data_received_256 446
 - max_data_received_31999 452
 - max_data_received_4096 450
 - max_data_received_512 447
 - max_data_received_64000 453
 - max_data_received_8192 451
 - max_data_received_gt64000 454
 - outbound_bytes_received 443
 - outbound_bytes_received_bottom 445
 - outbound_bytes_received_top 444
- outbound bytes sent
 - outbound_bytes_sent 442
 - outbound_bytes_sent_bottom 444
 - outbound_bytes_sent_top 444
- outbound communication
 - outbound_appl_id 195
 - outbound_comm_address 441
 - outbound_comm_protocol 441
- outbound sequence
 - outbound_sequence_no 196
- overflow records
 - first_overflow_time element 411
 - last_over_flow time element 411
- overflow_accesses 353
- package cache 280
 - pkg_cache_inserts 281
 - pkg_cache_lookups 280
 - pkg_cache_num_overflow 282
 - pkg_cache_size_top 282
- package names
 - package_name element 379
- packages
 - package_version_id element 380
- pages
 - data_object_pages element 356
- parallelism
 - degree_parallelism 403
- participant_no 312
- partition information
 - partition_number monitor element 416
- partitions
 - coord_partition_num 482
- pass-through
 - passthru 468
 - passthru_time 472
- pool_cur_size 222

monitor elements (*continued*)

- pool_id 220
- pool_max_size 223
- pool_secondary_id 221
- pool_watermark 223
- prefetching
 - unread_prefetch_pages element 269
- priv_workspace_num_overflows element 287
- progress_work_metric element 420
- queries
 - query_card_estimate 387
 - query_cost_estimate 388
 - queue_assignments_total 491
 - queue_size_top 491
 - queue_time_total 492
 - select_time 469
- quiescer
 - quiescer_auth_id 339
 - quiescer_obj_id 340
 - quiescer_state 340
 - quiescer_ts_id 340
- ranges
 - bottom 478
 - range_adjustment element 345
 - range_container_id element 346
 - range_end_stripe element 345
 - range_max_extent element 344
 - range_max_page_number element 344
 - range_num_containers 346
 - range_number element 344
 - range_offset element 346
 - range_start_stripe element 345
 - range_stripe_set_number element 344
- Real-time statistics
 - overview 506
- rebinding
 - int_auto_rebinds element 373
- records
 - partial_record element 412
- rej_curs_blk 365
- reoptimization
 - stmt_value_isreopt 395
- reorg_completion element 362
- reorg_long_tbspc_id 363
- reorg_tbspc_id 363
- reorganization
 - page_reorgs element 356
 - reorg_current_counter element 361
 - reorg_max_phase element 361
 - reorg_phase monitor element 360
 - reorg_phase_start element 360
 - reorg_rows_compressed monitor element 363
 - reorg_rows_rejected_for_compression monitor element 364
 - reorg_start element 362
 - reorg_status element 359
 - reorg_type element 359
- request_exec_time_avg 485
- response time
 - delete_time element 471
 - host_response_time element 459
 - insert_time element 470
- roll-forward recovery
 - rf_log_num 324
 - rf_status 324
 - rf_timestamp 323
 - rf_type 323

- monitor elements (*continued*)
 - rollback
 - rollback_sql_stmts 370
 - rolled_back_appl_id 322
 - rolled_back_participant_no 313
 - rolled_back_sequence_no 322
 - rollbacks
 - int_rollbacks 374
 - rollforward recovery
 - overview 323
 - rows
 - int_rows_inserted element 355
 - int_rows_updated element 354
 - rows_deleted element 349
 - rows_fetched 492
 - rows_inserted element 349
 - rows_modified 492
 - rows_read element 352
 - rows_returned 493
 - rows_selected element 350
 - rows_updated element 350
 - rows_written element 351
 - sp_rows_selected element 469
 - RUNSTATS utility
 - async_runstats 508
 - sync_runstats 507
 - sync_runstats_time 509
 - section_env 495
 - sections
 - priv_workspace_section_inserts element 288
 - priv_workspace_section_lookups element 287
 - section_number element 381
 - sequences
 - progress_seq_num element 419
 - sequence_no 191
 - server identification and status 172
 - servers
 - product_name 175
 - server_instance_name 172
 - server_platform 175
 - server_prdid 173
 - server_version 174
 - service levels
 - service_level 174
 - shared workspace
 - shr_workspace_num_overflows 284
 - shr_workspace_section_inserts 285
 - shr_workspace_section_lookups 285
 - shr_workspace_size_top 283
 - snapshot monitor logical data groups 115
 - snapshot monitoring 408
 - snapshots
 - time_stamp element 410
 - sort 224
 - sorting
 - piped_sorts_accepted element 226
 - piped_sorts_requested element 226
 - post_shrthreshold_sorts monitor element 225
 - post_threshold_sorts element 224
 - sort_heap_allocated element 224
 - sort_heap_top monitor element 230
 - sort_overflows element 228
 - sort_shrheap_allocated monitor element 230
 - sort_shrheap_top monitor element 230
 - total_sorts element 227
 - SQL communications area (SQLCA)
 - sqlca 387
- monitor elements (*continued*)
 - SQL cursors 364
 - SQL operations
 - elapsed_exec_time element 458
 - SQL statement activity 367
 - SQL statement details 377
 - SQL statements
 - ddl_sql_stmts element 372
 - dynamic_sql_stmts element 368
 - failed_sql_stmts element 368
 - insert_sql_stmts element 465
 - num_compilation element 401
 - num_executions element 401
 - select_sql_stmts element 371
 - sql_chains 438
 - sql_reqs_since_commit element 376
 - sql_stmts 437
 - static_sql_stmts element 367
 - stmt_pkgcache_id element 393
 - stmt_query_id element 392
 - stmt_sorts element 385
 - stmt_source_id element 392
 - stmt_text element 385
 - stmt_value_data element 394
 - stmt_value_index element 395
 - stmt_value_isnull element 394
 - stmt_value_type element 394
 - total_exec_time element 402
 - uid_sql_stmts element 371
 - SQL workspaces 283
 - statements
 - prep_time_best element 402
 - prep_time_worst element 402
 - stmt_first_use_time element 389
 - stmt_history_id element 389
 - stmt_history_list_size element 395
 - stmt_invocation_id element 391
 - stmt_isolation element 390
 - stmt_last_use_time 390
 - stmt_nest_level element 391
 - stmt_node_number element 376
 - stmt_type element 378
 - status
 - db_status element 179
 - db2_status element 175
 - dcs_appl_status element 439
 - ss_status element 397
 - storage paths
 - num_db_storage_paths 181
 - stored procedures
 - stored_proc_time element 473
 - stored_procs element 468
 - stripe sets
 - container_stripe_set monitor element 343
 - subsection details 396
 - table activity 346
 - table reorganization 359
 - reorg_end element 362
 - table spaces
 - activity 324
 - quiescer activity 339
 - range status 343
 - tablespace_auto_resize_enabled 336
 - tablespace_content_type 326
 - tablespace_cur_pool_id 328
 - tablespace_current_size 337
 - tablespace_extent_size 328

monitor elements (*continued*)

- table spaces (*continued*)
 - tablespace_free_pages 330
 - tablespace_id 324
 - tablespace_increase_size 337
 - tablespace_increase_size_percent 338
 - tablespace_initial_size 336
 - tablespace_last_resize_failed 339
 - tablespace_last_resize_time 338
 - tablespace_max_size 337
 - tablespace_min_recovery_time 334
 - tablespace_name 325
 - tablespace_next_pool_id 329
 - tablespace_num_containers 335
 - tablespace_num_quiescers 333
 - tablespace_num_ranges 335
 - tablespace_page_size 327
 - tablespace_page_top 331
 - tablespace_pending_free_pages 330
 - tablespace_prefetch_size 328
 - tablespace_rebalancer_extents_processed 332
 - tablespace_rebalancer_extents_remaining 332
 - tablespace_rebalancer_last_extent_moved 333
 - tablespace_rebalancer_mode 331
 - tablespace_rebalancer_priority 333
 - tablespace_rebalancer_restart_time 332
 - tablespace_rebalancer_start_time 331
 - tablespace_state 326
 - tablespace_state_change_object_id 334
 - tablespace_state_change_ts_id 334
 - tablespace_total_pages 329
 - tablespace_type 326
 - tablespace_usable_pages 329
 - tablespace_used_pages 330
 - tablespace_using_auto_storage 336
 - ts_name 323
- tables
 - table_file_id element 355
 - table_name element 347
 - table_schema element 348
 - table_type element 346
- territory_code 199
- thresholds
 - num_threshold_violations 489
 - threshold_action 497
 - threshold_domain 498
 - threshold_maxvalue 498
 - threshold_name 499
 - threshold_predicate 499
 - threshold_queuesize 499
 - thresholdid 500
- time
 - prefetch_wait_time element 269
 - prep_time 491
 - progress_start_time element 420
 - ss_exec_time element 397
 - stmt_elapsed_time element 384
 - time_completed 500
 - time_created 501
 - time_of_violation 501
 - time_started 501
 - total_sort_time element 227
- time zones
 - time_zone_disp element 176
- timestamps
 - activate_timestamp 474
 - db_conn_time 178

monitor elements (*continued*)

- timestamps (*continued*)
 - db2start_time 172
 - last_backup 181
 - last_reset 409
 - lock_wait_start_time 319
 - message_time 416
 - statistics_timestamp 496
 - status_change_time 187
 - stmt_start 382
 - stmt_stop 383
- tokens
 - consistency_token monitor element 380
 - corr_token monitor element 197
- total_hash_loops element 233
- tq_cur_send_spills 399
- tq_id_waiting_on 401
- tq_max_send_spills 400
- tq_node_waited_for 398
- tq_rows_read 399
- tq_rows_written 400
- tq_tot_send_spills 398
- tq_wait_for_any 398
- transaction processing
 - tpmon_acc_str 464
 - tpmon_client_userid 463
 - tpmon_client_wkstn 463
- transaction processor monitoring 462
- transactions
 - num_indoubt_trans 316
 - xid monitor element 458
- units of work (UOW)
 - prev_uow_stop_time 205
 - progress_total_units element 420
 - uow_comp_status 207
 - uow_elapsed_time 207
 - uow_id 502
 - uow_start_time 205
 - uow_status 207
 - uow_stop_time 206
- updates
 - update_sql_stmts element 466
- utilities 416
 - utility_description element 418
 - utility_id element 416
 - utility_invoker_type element 418
 - utility_priority element 417
 - utility_type 417
- vectored_ios 270
- watermarks
 - concurrent_act_top 478
 - concurrent_connection_top 479
 - concurrent_wlo_act_top 479
 - concurrent_wlo_top 480
 - coord_act_lifetime_top 481
 - cost_estimate_top 482
 - rows_returned_top 493
 - temp_tablespace_top 497
- Workload management
 - overview 474
- workloads
 - wlo_completed_total 503
 - workload_id 504
 - workload_name 505
 - workload_occurrence_id 505
- xquery_stmts 377

- monitor heap
 - health indicators
 - db2.mon_heap_util 574
- monitor switches
 - description 21
 - setting from a client application 5
 - setting from the CLP 7
- monitoring
 - buffer pool efficiency
 - administrative views 86
 - capturing a snapshot from client applications 38
 - capturing a snapshot from the command line 35
 - capturing a snapshot using SQL 26, 34
 - with file access 30
 - with SNAP_WRITE_FILE 28
 - data partitions 88
 - database activity 81, 87
 - database events 49
 - event types 49
 - overview 3
 - sample output 68
 - database recovery 323
 - databases
 - overview 3
 - health monitor 515, 525
 - memory components 99
 - open access to monitor data
 - capturing snapshot information to a file 28
 - retrieving snapshot information from a file 30
 - SYSMON authority 25
 - runtime rollback process 87
 - snapshot
 - API request types 39
 - CLP commands 35
 - overview 3
 - system monitor 3
 - with db2top 16
- most recent response time for connect monitor element 461

N

- names
 - monitor elements
 - db_name element 176
 - dcs_db_name element 434
 - service_subclass_name 495
 - service_superclass_name 496
 - work_action_set_name 503
 - work_class_name 504
- network time
 - monitor elements
 - max_network_time_1_ms 454
 - max_network_time_100_ms 456
 - max_network_time_16_ms 455
 - max_network_time_4_ms 455
 - max_network_time_500_ms 456
 - max_network_time_gt500_ms 457
 - network_time_bottom 458
 - network_time_top 457
- nicknames
 - health indicator 575
 - monitor elements
 - create_nickname element 467
 - create_nickname_time element 472
- nodes
 - monitor elements
 - coord_node monitor element 203

- nodes (*continued*)
 - monitor elements (*continued*)
 - node_number element 202
 - num_nodes_in_db2_instance 410
 - ss_node_number element 396
- notices 593
- num_indoubt_trans element 316
- num_log_data_found_in_buffer element 299
- num_log_part_page_io element 298
- num_log_read_io element 297
- num_log_write_io element 297
- num_transmissions element 460
- num_transmissions_group element 460
- num_vectored_IOs element 270
- numbers
 - monitor elements
 - progress_list_cur_seq_num element 419
 - ss_number element 396

O

- objects
 - performance on Windows 106
- OLAP functions
 - monitor elements 235, 236, 237
- Online Analytical Processing (OLAP)
 - monitor elements
 - overview 234
- operation monitor element 378
- operations
 - monitor elements
 - direct_read_reqs element 274
 - direct_read_time element 275
 - direct_reads element 273
 - direct_write_reqs element 275
 - direct_write_time element 276
 - direct_writes element 273
 - stmt_operation element 378
- optimization
 - monitor elements
 - stmt_value_isreopt 395
- ordering DB2 books 586
- outbound bytes received
 - monitor elements
 - max_data_received_1024 448
 - max_data_received_128 445
 - max_data_received_16384 451
 - max_data_received_2048 449
 - max_data_received_256 446
 - max_data_received_31999 452
 - max_data_received_4096 450
 - max_data_received_512 447
 - max_data_received_64000 453
 - max_data_received_8192 451
 - max_data_received_gt64000 454
 - outbound_bytes_received 443
 - outbound_bytes_received_bottom 445
 - outbound_bytes_received_top 444
- outbound bytes sent
 - monitor elements
 - max_data_sent_1024 448
 - max_data_sent_128 445
 - max_data_sent_16384 451
 - max_data_sent_2048 448
 - max_data_sent_256 446
 - max_data_sent_31999 452
 - max_data_sent_4096 449

- outbound bytes sent *(continued)*
 - monitor elements *(continued)*
 - max_data_sent_512 447
 - max_data_sent_64000 453
 - max_data_sent_8192 450
 - max_data_sent_gt64000 454
 - outbound_bytes_sent 442
 - outbound_bytes_sent_bottom 444
 - outbound_bytes_sent_top 444
- outbound communication
 - monitor elements
 - outbound_appl_id 195
 - outbound_comm_address 441
 - outbound_comm_protocol 441
 - outbound_sequence_no 196
- overflow records
 - monitor elements
 - first_overflow_time element 411
 - last_over_flow time element 411

P

- package cache
 - db.pkgcache_hitratio 573
 - monitor elements
 - pkg_cache_inserts 281
 - pkg_cache_lookups 280
 - pkg_cache_num_overflow 282
 - pkg_cache_size_top 282
- packages
 - names
 - package_name monitor element 379
 - package_version_id monitor element 380
- pages
 - bp_pages_left_to_remove monitor element 272
 - data_object_pages monitor element 356
 - removing
 - bp_pages_left_to_remove monitor element 272
- parallelism
 - monitor elements
 - degree_parallelism 403
 - intra-query 403
- partial_record monitor element 412
- partition_number monitor element 416
- partitioned database environments
 - coord_partition_num monitor element 482
 - event monitoring 66
 - global snapshots 44
- partitioned tables
 - reorganizing 88
- pass-through
 - monitor elements
 - passthru_time 472
 - passthru 468
- performance
 - information
 - displaying 106
 - enabling remote access 106
 - remote databases 107
 - resetting values 108
 - Windows
 - monitoring tools 105
 - Performance Monitor objects 106
- pipe event monitors
 - creating 64
 - formatting output from command line 75
 - named pipe management 65

- pipedsorts_accepted monitor element 226
- pipedsorts_requested monitor element 226
- post_shrthreshold_sorts monitor element 225
- post_threshold_sorts monitor element 224
- prefetching
 - monitor elements
 - unread_prefetch_pages 269
- priv_workspace_num_overflows monitor element
 - description 287
- priv_workspace_section_inserts monitor element
 - description 288
- priv_workspace_section_lookups monitor element
 - description 287
- priv_workspace_size_top monitor element
 - description 286
- problem determination
 - information available 590
 - tutorials 590
- processes
 - monitor elements
 - agent_pid 208
- progress_description monitor element 419
- progress_seq_num monitor element 419
- progress_start_time monitor element 420
- progress_work_metric monitor element 420

Q

- queries
 - monitor elements
 - query_card_estimate 387
 - query_cost_estimate 388
 - queue_assignments_total 491
 - queue_size_top 491
 - queue_time_total 492
 - select_time 469
- quiescer
 - monitor elements
 - quiescer_auth_id 339
 - quiescer_obj_id 340
 - quiescer_state 340
 - quiescer_ts_id 340

R

- range adjustment monitor element 345
- range container monitor element 346
- range number monitor element 344
- range offset monitor element 346
- range_num_containers monitor element 346
- ranges
 - monitor elements
 - bottom 478
 - range_adjustment element 345
 - range_container_id element 346
 - range_end_stripe element 345
 - range_max_extent element 344
 - range_max_page_number element 344
 - range_num_containers 346
 - range_number element 344
 - range_offset element 346
 - range_start_stripe element 345
 - range_stripe_set_number element 344
- real-time statistics
 - monitor elements
 - overview 506

- real-time statistics (*continued*)
 - monitor elements (*continued*)
 - stats_fabricate_time 508
 - stats_fabrications 506
- rebalancing
 - monitor elements
 - tablespace_rebalancer_extents_processed 332
 - tablespace_rebalancer_extents_remaining 332
 - tablespace_rebalancer_last_extent_moved 333
 - tablespace_rebalancer_mode 331
 - tablespace_rebalancer_priority 333
 - tablespace_rebalancer_restart_time 332
 - tablespace_rebalancer_start_time 331
- rebinding
 - monitor elements
 - int_auto_rebinds element 373
- records
 - monitor elements
 - partial_record element 412
- recovery
 - monitor elements
 - log_to_redo_for_recovery 296
- remote
 - performance 107
- reoptimization
 - monitor elements
 - stmt_value_isreopt 395
- reorg_index_id monitor element 363
- reorganization
 - health indicators
 - db.tb_reorg_req 567
 - monitor elements
 - page_reorgs element 356
 - reorg_current_counter element 361
 - reorg_max_counter element 361
 - reorg_max_phase element 361
 - reorg_phase monitor element 360
 - reorg_phase_start element 360
 - reorg_rows_compressed monitor element 363
 - reorg_rows_rejected_for_compression monitor element 364
 - reorg_start element 362
 - reorg_status element 359
 - reorg_type element 359
- reorganize phase monitor element 360
- reporting 68
- request identifier for sql statement monitor element 415
- response time
 - monitor elements
 - delete_time element 471
 - host_response_time element 459
 - insert_time element 470
- roll-forward recovery
 - monitor elements
 - overview 323
 - rf_log_num 324
 - rf_status 324
 - rf_timestamp 323
 - rf_type 323
 - tablespace_min_recovery_time 334
 - ts_name 323
- rollbacks
 - monitor elements
 - int_deadlock_rollbacks 375
 - int_rollbacks 374
 - rf_status 324
 - rollback_sql_stmts 370
- rollbacks (*continued*)
 - monitor elements (*continued*)
 - rolled_back_agent_id 322
 - rolled_back_appl_id 322
 - rolled_back_participant_no 313
 - rolled_back_sequence_no 322
 - monitoring progress 87
- rows
 - monitor elements
 - int_rows_inserted 355
 - int_rows_updated 354
 - rows_deleted 349
 - rows_fetched 492
 - rows_inserted 349
 - rows_modified 492
 - rows_read 352
 - rows_returned 493
 - rows_returned_top 493
 - rows_selected 350
 - rows_updated 350
 - rows_written 351
 - sp_rows_selected 469
 - rows compressed monitor element 363
 - rows deleted monitor element 349
 - rows inserted monitor element 349
 - rows read monitor element 352
 - rows rejected for compression monitor element 364
 - rows returned by stored procedures monitor element 469
 - rows selected monitor element 350
 - rows updated monitor element 350
 - rows written monitor element 351
- RUNSTATS utility
 - monitor elements
 - async_runstats 508
 - sync_runstats 507
 - sync_runstats_time 509

S

- schemas
 - tables
 - table_schema element 348
- secondary logs allocated currently monitor element 292
- section number monitor element 381
- sections
 - monitor elements
 - appl_section_inserts 289
 - appl_section_lookups 289
 - priv_workspace_section_inserts element 288
 - priv_workspace_section_lookups element 287
 - section_number element 381
- select SQL statements executed monitor element 371
- self-describing data stream
 - database system monitor 10
 - event monitors 76
 - snapshot monitor 45
- sequences
 - monitor elements
 - progress_seq_num element 419
 - sequence_no 191
 - sequence_no_holding_lk 321
- servers
 - monitor elements
 - product_name 175
 - server_instance_name 172
 - server_platform 175
 - server_prdid 173

- servers *(continued)*
 - monitor elements *(continued)*
 - server_version 174
- service-level information
 - monitor elements
 - service_level 174
- session authorization ID 193
- shared workspace
 - health indicators
 - db.shrworkspace_hitratio 573
 - monitor elements
 - shr_workspace_num_overflows 284
 - shr_workspace_section_inserts 285
 - shr_workspace_section_lookups 285
 - shr_workspace_size_top 283
- snapshot monitoring
 - administrative views 84
 - API request types 39
 - capturing
 - using SQL with file access 30
 - capturing to file 28
 - CLP commands 35
 - description 25
 - interpreting output for data partitions 88
 - making snapshot data available for all users 28
 - on data partitions 88
 - on partitioned database systems 44
 - output
 - samples 41
 - self-describing data stream 45
 - overview 3
 - request types 35
 - SQL table functions 31
 - subsections
 - subsection snapshots 43
 - using a client application 38
 - using CLP 35
 - using SQL 34
 - using SQL with direct access 26
 - with SNAP_WRITE_FILE 28
- snapshot time monitor element 410
- snapshots
 - capturing
 - using SQL with file access 30
 - capturing to file 28
 - capturing with SNAP_WRITE_FILE 28
 - making snapshot data available for all users 28
 - monitor elements
 - time_stamp element 410
 - SQL table functions 31
 - using SQL with direct access 26
- sort overflows monitor element 228
- Sort Private Heap High Watermark monitor element 230
- sort share heap currently allocated monitor element 230
- Sort Share Heap High Watermark monitor element 230
- sorting
 - health indicators
 - db2.sort_privmem_util 562
 - monitor elements 224
 - active_sorts 229
 - db.spilled_sorts 564
 - pipd_sorts_accepted element 226
 - pipd_sorts_requested element 226
 - post_shrthreshold_sorts monitor element 225
 - post_threshold_sorts element 224
 - sort_heap_allocated element 224
 - sort_heap_top monitor element 230
- sorting *(continued)*
 - monitor elements *(continued)*
 - sort_overflows element 228
 - sort_shrheap_allocated monitor element 230
 - sort_shrheap_top monitor element 230
 - total_sorts element 227
- SQL communications area (SQLCA)
 - monitor elements
 - sqlca 387
- SQL dynamic statement text monitor element 385
- SQL operations
 - monitor elements
 - elapsed_exec_time element 458
- SQL requests since last commit monitor element 376
- SQL statements
 - displaying help 587
 - monitor elements
 - ddl_sql_stmts element 372
 - dynamic_sql_stmts element 368
 - failed_sql_stmts element 368
 - insert_sql_stmts element 465
 - num_compilation element 401
 - num_executions element 401
 - select_sql_stmts element 371
 - sql_chains 438
 - sql_reqs_since_commit element 376
 - sql_stmts 437
 - static_sql_stmts element 367
 - stmt_pkgcache_id element 393
 - stmt_query_id element 392
 - stmt_sorts element 385
 - stmt_source_id element 392
 - stmt_text element 385
 - stmt_value_data element 394
 - stmt_value_index element 395
 - stmt_value_isnull element 394
 - stmt_value_type element 394
 - total_exec_time element 402
 - uid_sql_stmts element 371
- SQL table functions
 - capturing health snapshots 526
 - health monitor 578
- SQL workspaces
 - monitor elements 283
- sql_chains element 438
- sql_stmts element 437
- start stripe monitor element 345
- statement best preparation time monitor element 402
- statement first use time monitor element 389
- statement history identifier monitor element 389
- statement history list size monitor element 395
- statement invocation identifier monitor element 391
- statement isolation monitor element 390
- statement last use time monitor element 390
- statement nesting level monitor element 391
- statement node monitor element 376
- statement operation monitor element 378
- statement package cache identifier monitor element 393
- statement query identifier monitor element 392
- statement sorts monitor element 385
- statement source identifier monitor element 392
- statement type monitor element 378
- statement worst preparation time monitor element 402
- statements
 - monitor elements
 - prep_time_best element 402
 - prep_time_worst element 402

- statements (*continued*)
 - monitor elements (*continued*)
 - stmt_first_use_time element 389
 - stmt_history_id element 389
 - stmt_history_list_size element 395
 - stmt_invocation_id element 391
 - stmt_isolation element 390
 - stmt_last_use_time 390
 - stmt_nest_level element 391
 - stmt_node_number element 376
 - stmt_type element 378
- STATEMENTS event type
 - overview 49
- states
 - health indicators
 - db.alert_state 566
 - db.db_op_status 566
 - db2.db2_op_status 565
 - ts.ts_op_status 562
- static SQL statements attempted monitor element 367
- statistics collection
 - health indicators
 - db.tb_runstats_req 567
- status
 - monitor elements
 - appl_status 185
 - db_status element 179
 - db2_status element 175
 - dcs_appl_status element 439
 - ss_status element 397
- stmt_operation element 378
- storage paths
 - monitor elements
 - num_db_storage_paths 181
- stored procedure time monitor element 473
- stored procedures
 - monitor elements
 - stored_proc_time element 473
 - stored_procs element 468
- stored procedures monitor element 468
- stripe set number monitor element 344
- stripe sets
 - monitor elements
 - container_stripe_set monitor element 343
- subsection execution elapsed time monitor element 397
- subsection node number monitor element 396
- subsection number monitor element 396
- subsection snapshots 43
- subsection status monitor element 397
- subsections
 - snapshot monitoring
 - subsection snapshots 43
- summary
 - health indicators 555
- SYSMON authority 25
- system monitor 3
- system monitor switches
 - description 21
 - setting from a client application 5
 - setting from the CLP 7
 - types 21
- System monitoring guide and reference
 - overview xi

T

- table event monitors
 - creating 53
 - table management 56
- table file ID monitor element 355
- table name monitor element 347
- table queues
 - monitor elements
 - tq_cur_send_spills 399
 - tq_id_waiting_on 401
 - tq_max_send_spills 400
 - tq_node_waited_for 398
 - tq_rows_read 399
 - tq_rows_written 400
 - tq_tot_send_spills 398
 - tq_wait_for_any 398
- table reorganization
 - monitor elements 359
 - reorg_end element 362
- table reorganize attribute flag monitor element 359
- table reorganize completion flag monitor element 362
- table reorganize end time monitor element 362
- table reorganize phase start time monitor element 360
- table reorganize start time monitor element 362
- table reorganize status monitor element 359
- table schema name monitor element 348
- table spaces
 - health indicators
 - ts.ts_auto_resize_status 559
 - ts.ts_op_status 562
 - ts.ts_util 560
 - ts.ts_util_auto_resize 560
 - tsc.tscont_op_status 562
 - tsc.utilization 561
 - monitor elements
 - activity 324
 - bp_tbsp_use_count 272
 - quiescer activity 339
 - quiescer_ts_id 340
 - range status 343
 - reorg_long_tbsp_id 363
 - reorg_tbsp_id 363
 - tablespace_auto_resize_enabled 336
 - tablespace_content_type 326
 - tablespace_cur_pool_id 328
 - tablespace_current_size 337
 - tablespace_extent_size 328
 - tablespace_free_pages 330
 - tablespace_id 324
 - tablespace_increase_size 337
 - tablespace_increase_size_percent 338
 - tablespace_initial_size 336
 - tablespace_last_resize_failed 339
 - tablespace_last_resize_time 338
 - tablespace_max_size 337
 - tablespace_min_recovery_time 334
 - tablespace_name 325
 - tablespace_next_pool_id 329
 - tablespace_num_containers 335
 - tablespace_num_quiescers 333
 - tablespace_num_ranges 335
 - tablespace_page_size 327
 - tablespace_page_top 331
 - tablespace_pending_free_pages 330
 - tablespace_prefetch_size 328
 - tablespace_rebalancer_extents_processed 332
 - tablespace_rebalancer_extents_remaining 332

- table spaces (*continued*)
 - monitor elements (*continued*)
 - tablespace_rebalancer_last_extent_moved 333
 - tablespace_rebalancer_mode 331
 - tablespace_rebalancer_priority 333
 - tablespace_rebalancer_restart_time 332
 - tablespace_rebalancer_start_time 331
 - tablespace_state 326
 - tablespace_state_change_object_id 334
 - tablespace_state_change_ts_id 334
 - tablespace_total_pages 329
 - tablespace_type 326
 - tablespace_usable_pages 329
 - tablespace_used_pages 330
 - tablespace_using_auto_storage 336
 - ts_name 323
 - table type monitor element 346
- tables
 - monitor elements
 - table_file_id element 355
 - table_name element 347
 - table_schema element 348
 - table_type element 346
 - schemas
 - table_schema element 348
- TABLES event type
 - overview 49
- TABLESPACES event type
 - overview 49
- terms and conditions
 - use of publications 590
- territory codes
 - monitor elements
 - territory_code 199
- threads
 - monitor elements
 - agent_pid 208
- thresholds
 - monitor elements
 - num_threshold_violations 489
 - threshold_action 497
 - threshold_domain 498
 - threshold_maxvalue 498
 - threshold_name 499
 - threshold_predicate 499
 - threshold_queuesize 499
 - thresholdid 500
 - threshold-based health indicators 515, 551
- time
 - monitor elements
 - prefetch_wait_time element 269
 - prep_time 491
 - progress_start_time element 420
 - ss_exec_time element 397
 - stmt_elapsed_time element 384
 - time_completed 500
 - time_created 501
 - time_of_violation 501
 - time_started 501
 - total_sort_time element 227
 - time waited for prefetch monitor element 269
 - time zone displacement monitor element 176
- time zones
 - monitor elements
 - time_zone_disp element 176
- timestamps
 - monitor elements
 - activate_timestamp 474
 - db_conn_time 178
 - db2start_time 172
 - last_backup 181
 - last_reset 409
 - lock_wait_start_time 319
 - message_time 416
 - prev_uow_stop_time 205
 - statistics_timestamp 496
 - status_change_time 187
 - stmt_start 382
 - stmt_stop 383
 - uow_start_time 205
 - uow_stop_time 206
- tokens
 - monitor elements
 - consistency_token monitor element 380
 - corr_token monitor element 197
 - total fcm buffers received monitor element 238
 - total fcm buffers sent monitor element 238
 - total hash loops monitor element 233
 - total log available monitor element 294
 - total log space used monitor element 294
 - total number of pages in object monitor element 361
 - total progress work units monitor element 420
 - total sort heap allocated monitor element 224
 - total sort time monitor element 227
 - total sorts monitor element 227
 - transaction event monitor 68
 - transaction ID monitor element 458
 - transaction processing monitors
 - monitor elements
 - tpmon_acc_str 464
 - tpmon_client_app 464
 - tpmon_client_userid 463
 - tpmon_client_wkstn 463
- transactions
 - monitor elements
 - num_indoubt_trans 316
 - xid monitor element 458
- TRANSACTIONS event type
 - overview 49
- troubleshooting
 - online information 590
 - tutorials 590
- tutorials
 - problem determination 590
 - troubleshooting 590
 - Visual Explain 590
- type at monitored (server) node monitor element 173

U

- units of work (UOW)
 - monitor elements
 - prev_uow_stop_time 205
 - progress_total_units element 420
 - uow_comp_status 207
 - uow_elapsed_time 207
 - uow_id 502
 - uow_lock_wait_time 319
 - uow_log_space_used 293
 - uow_start_time 205
 - uow_status 207
 - uow_stop_time 206

- unread prefetch pages monitor element 269
- update response time monitor element 471
- update_time element 471
- update/insert/delete SQL statements executed monitor element 371
- updates
 - DB2 Information Center 588
 - monitor elements
 - update_sql_stmts element 466
- updates monitor element 466
- user authorization level monitor element
 - authorizations
 - authority_lvl element 200
- utilities
 - monitor elements
 - utility_description element 418
 - utility_id element 416
 - utility_invoker_type element 418
 - utility_priority element 417
 - utility_type 417
- utility description monitor element 418
- utility id monitor element 416
- utility invoker type monitor element 418
- utility priority monitor element 417
- utility start time monitor element 417
- utility state monitor element 418
- utility_dbname element 416
- utility_start_time element 417
- utility_state element 418

V

- value data monitor element 394
- value has null value monitor element 394
- value index monitor element 395
- value type monitor element 394
- version monitor element 412
- version of monitor data monitor element 412
- Visual Explain
 - tutorial 590

W

- watermark monitor elements
 - concurrent_act_top 478
 - concurrent_connection_top 479
 - concurrent_wlo_act_top 479
 - concurrent_wlo_top 480
 - coord_act_lifetime_top 481
 - cost_estimate_top 482
 - rows_returned_top 493
- watermarks
 - monitor elements
 - temp_tablespace_top 497
- Windows Management Instrumentation (WMI)
 - DB2 database system integration 103
 - description 103
- Windows operating systems
 - Performance Monitor
 - description 105
 - registering DB2 105
- WMI (Windows Management Instrumentation)
 - see Windows Management Instrumentation (WMI) 103
- workload management
 - monitor elements 474

- workloads
 - monitor elements
 - wlo_completed_total 503
 - workload_id 504
 - workload_name 505
 - workload_occurrence_id 505
- write-to-table event monitors
 - buffering 63

X

- XDA Object Pages monitor element 358
- xda_object_pages monitor element 358
- xquery_stmts monitor element 377



Printed in USA

SC23-5865-02



Spine information:

DB2 Version 9.5 for Linux, UNIX, and Windows

System Monitor Guide and Reference

