

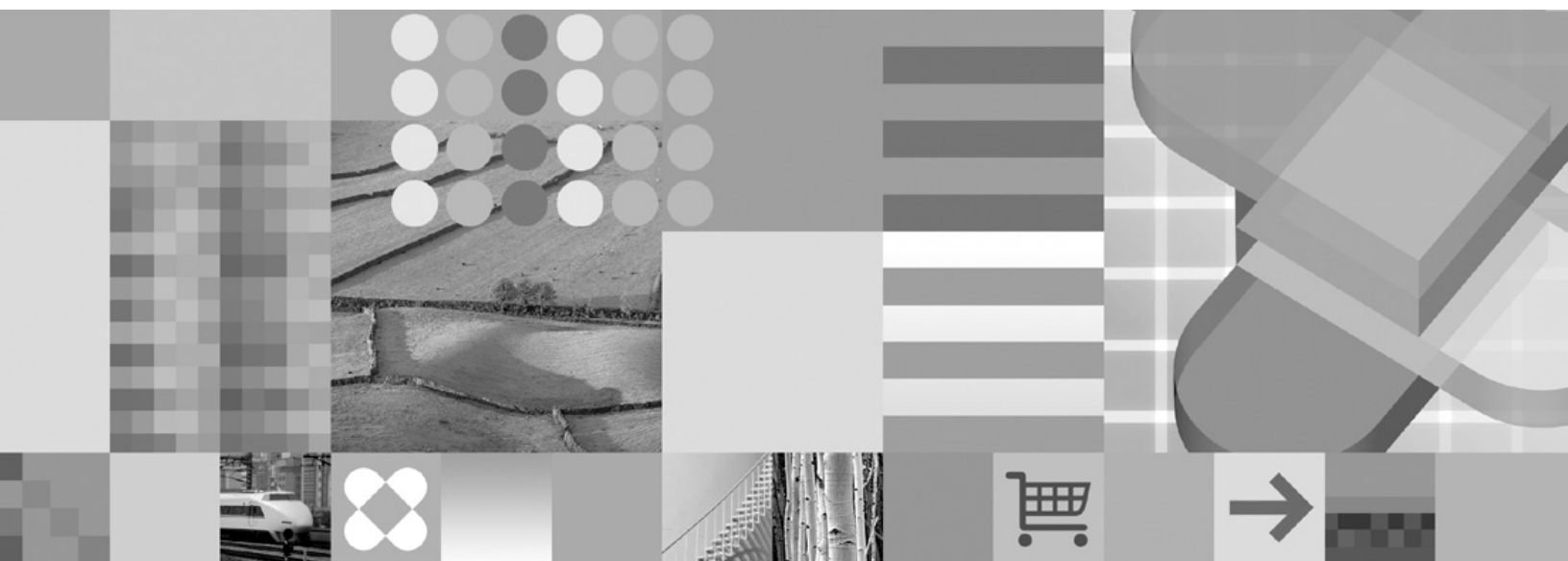


System Monitor Guide and Reference

DB2®



DB2 Version 9
for Linux, UNIX, and Windows



System Monitor Guide and Reference

Before using this information and the product it supports, be sure to read the general information under *Notices*.

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, 2006. All rights reserved.

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

Contents

Part 1. System Monitor Guide 1

Chapter 1. Introducing the Database System Monitor 3

Database system monitor	3
Database system monitor data organization	3
Comparison of DB2 monitors	5
Counter status and visibility	7
System monitor output: the self-describing data stream	8
Database system monitor memory requirements	9

Chapter 2. System Monitor Switches 13

System monitor switches	13
Setting monitor switches from the CLP	15
Setting monitor switches from a client application	17
Monitor switches self-describing data stream	19

Chapter 3. Using the Snapshot Monitor 21

Snapshot monitor	21
Access to system monitor data: SYSMON authority	22
Capturing database system snapshots using snapshot administrative views and table functions	22
Capturing database system snapshot information to a file using the SNAP_WRITE_FILE stored procedure	25
Accessing database system snapshots using snapshot table functions in SQL queries (with file access)	27
Using snapshot monitor data to monitor the reorganization of a partitioned table	29
Snapshot monitor SQL Administrative Views	37
Scenario: Identifying costly applications using snapshot administrative views	40
Scenario: Monitoring buffer pool efficiency using administrative views	42
Capturing a database snapshot from the CLP	43
Snapshot monitor CLP commands	44
Capturing a database snapshot from a client application	46
Snapshot monitor API request types	48
Snapshot monitor sample output	50
Subsection snapshots	52
Global snapshots on partitioned database systems	53
Snapshot monitor self-describing data stream	54
SQL access to database system snapshots	57

Chapter 4. Using Event Monitors 59

Event monitors	59
Event types	60
Collecting information about database system events	61
Creating an event monitor	63
Creating a table event monitor	64
Event monitor table management	67

Creating a file event monitor	71
Event monitor file management	73
Write-to-table and file event monitor buffering	74
Creating a pipe event monitor	75
Event monitor named pipe management	76
Creating an event monitor for partitioned databases	77
Formatting file or pipe event monitor output from a command line	79
Event monitor sample output	80
Event records and their corresponding applications	90
Event monitor self-describing data stream	91
Transferring event monitor data between systems	93

Part 2. System Monitor Reference 95

Chapter 5. System Monitor Logical Data Groups 97

Snapshot monitor interface mappings to logical data groups	97
Snapshot monitor logical data groups and monitor elements	100
Event type mappings to logical data groups	129
Event monitor logical data groups and monitor elements	131

Chapter 6. Monitor elements 147

Database system monitor elements	147
Server identification and status	148
Server identification and status monitor elements	148
db2start_time - Start Database Manager Timestamp	148
server_nname - Configuration NNAME at Monitoring (Server) Database Partition	149
server_instance_name - Server Instance Name	149
server_db2_type - Database Manager Type at Monitored (Server) Node	150
server_prdid - Server Product/Version ID	150
server_version - Server Version	151
service_level - Service Level	151
server_platform - Server Operating System	152
product_name - Product Name	152
db2_status - Status of DB2 Instance	152
time_zone_disp - Time Zone Displacement	153
Database identification and status	153
Database identification and status monitor elements	153
db_name - Database Name	154
db_path - Database Path	155
db_conn_time - Database Activation Timestamp	155
conn_time - Time of Database Connection	156
disconn_time - Database Deactivation Timestamp	156
db_status - Status of Database	156

catalog_node_name - Catalog Node Network Name	157	uow_start_time - Unit of Work Start Timestamp	185
db_location - Database Location	157	uow_stop_time - Unit of Work Stop Timestamp	185
catalog_node - Catalog Node Number	158	uow_elapsed_time - Most Recent Unit of Work Elapsed Time	186
last_backup - Last Backup Timestamp	158	uow_comp_status - Unit of Work Completion Status	186
num_db_storage_paths - Number of automatic storage paths	159	uow_status - Unit of Work Status	187
db_storage_path - Automatic storage path	159	appl_idle_time - Application Idle Time	187
sto_path_free_sz - Automatic Storage Path Free Space	159	data_partition_id - Data Partition Identifier monitor element	188
fs_used_size - Amount of Space Used on a File System	160	DB2 agent information	188
fs_total_size - Total Size of a File System	160	Database manager configuration	189
fs_id - Unique File System Identification Number	161	Database manager configuration monitor elements	189
fs_type - File System Type	162	Agents and connections	190
Application identification and status	162	Memory pool	202
Application identification and status monitor elements	162	Sort	207
agent_id - Application Handle (agent ID)	163	Hash join.	215
appl_status - Application Status	164	Fast communications manager.	219
codepage_id - ID of Code Page Used by Application	166	Utilities	222
status_change_time - Application Status Change Time	167	Database configuration	229
appl_id_oldest_xact - Application with Oldest Transaction	167	Database configuration monitor elements	229
smallest_log_avail_node - Node with Least Available Log Space	168	Buffer pool activity	229
appl_name - Application Name	168	Non-buffered I/O activity	268
appl_id - Application ID	169	Catalog cache	273
sequence_no - Sequence Number	172	Package cache	277
auth_id - Authorization ID	172	SQL workspaces	282
session_auth_id - Session Authorization ID	173	Database heap	288
client_nname - Configuration NNAME of Client	173	Logging	288
client_prdid - Client Product/Version ID	174	Database and application activity.	301
client_db_alias - Database Alias Used by Application	174	Database and application activity monitor elements	301
host_prdid - Host Product/Version ID	175	Locks and deadlocks	301
outbound_appl_id - Outbound Application ID	175	Lock wait information	317
outbound_sequence_no - Outbound Sequence Number	176	Rollforward monitoring	324
execution_id - User Login ID	176	Table space activity	327
corr_token - DRDA Correlation Token	177	Table activity	350
client_pid - Client Process ID	177	Table reorganization	363
client_platform - Client Operating Platform	178	SQL cursors	369
client_protocol - Client Communication Protocol	178	SQL statement activity	372
territory_code - Database Territory Code	179	SQL statement details	384
appl_priority - Application Agent Priority	179	Subsection details	403
appl_priority_type - Application Priority Type	180	Dynamic SQL	409
authority_lvl - User Authorization Level	180	Intra-query parallelism	411
node_number - Node Number	181	CPU usage	412
coord_node - Coordinating Node	182	Snapshot monitoring	419
appl_con_time - Connection Request Start Timestamp	183	Event monitoring	421
connections_top - Maximum Number of Concurrent Connections	183	High availability disaster recovery	427
conn_complete_time - Connection Request Completion Timestamp	184	High availability disaster recovery monitor elements	427
prev_uow_stop_time - Previous Unit of Work Completion Timestamp	184	hadr_role - HADR Role	427
		hadr_state - HADR State monitor element.	428
		hadr_syncmode - HADR Synchronization Mode monitor element	428
		hadr_connect_status - HADR Connection Status monitor element	429
		hadr_connect_time - HADR Connection Time monitor element	430
		hadr_heartbeat - HADR Heartbeat monitor element	430

hadr_local_host - HADR Local Host monitor element	431	outbound_bytes_received - Outbound Number of Bytes Received	450
hadr_local_service - HADR Local Service monitor element	432	inbound_bytes_sent - Inbound Number of Bytes Sent	450
hadr_remote_host - HADR Remote Host monitor element	432	outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent	451
hadr_remote_service - HADR Remote Service monitor element	433	outbound_bytes_received_top - Maximum Outbound Number of Bytes Received	451
hadr_remote_instance - HADR Remote Instance monitor element	433	outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent	452
hadr_timeout - HADR Timeout monitor element	433	outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received	452
hadr_primary_log_file - HADR Primary Log File monitor element	434	max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes	453
hadr_primary_log_page - HADR Primary Log Page monitor element	434	max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes	453
hadr_primary_log_lsn - HADR Primary Log LSN monitor element.	435	max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes	453
hadr_standby_log_file - HADR Standby Log File monitor element	435	max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes	454
hadr_standby_log_page - HADR Standby Log Page monitor element	436	max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes	454
hadr_standby_log_lsn - HADR Standby Log LSN monitor element.	436	max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes	455
hadr_log_gap - HADR Log Gap	437	max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes	455
DB2 Connect	437	max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes	456
DB2 Connect monitor elements	437	max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes	456
dcs_db_name - DCS Database Name	440	max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes	457
host_db_name - Host Database Name	440	max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes	457
gw_db_alias - Database Alias at the Gateway	440	max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes	458
gw_con_time - DB2 Connect Gateway First Connect Initiated	441	max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes	458
gw_connections_top - Maximum Number of Concurrent Connections to Host Database	441	max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes	459
gw_total_cons - Total Number of Attempted Connections for DB2 Connect	441	max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes	459
gw_cur_cons - Current Number of Connections for DB2 Connect	442	max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes	460
gw_cons_wait_host - Number of Connections Waiting for the Host to Reply	442		
gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request	443		
gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing	443		
sql_stmts - Number of SQL Statements Attempted	444		
sql_chains - Number of SQL Chains Attempted	444		
open_cursors - Number of Open Cursors	445		
dcs_appl_status - DCS Application Status	446		
agent_status - DCS Application Agents	446		
host_ccsid - Host Coded Character Set ID	447		
outbound_comm_protocol - Outbound Communication Protocol	447		
outbound_comm_address - Outbound Communication Address	448		
inbound_comm_address - Inbound Communication Address	448		
inbound_bytes_received - Inbound Number of Bytes Received	449		
outbound_bytes_sent - Outbound Number of Bytes Sent	449		

max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes	460
max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes	461
max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes	461
max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes	462
max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes	462
max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes	462
max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms	463
max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms	463
max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms	464
max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms	465
max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms	465
max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms	466
network_time_top - Maximum Network Time for Statement	466
network_time_bottom - Minimum Network Time for Statement	467
xid - Transaction ID	467
elapsed_exec_time - Statement Execution Elapsed Time	467
host_response_time - Host Response Time	468
num_transmissions - Number of Transmissions	469
num_transmissions_group - Number of Transmissions Group	469
con_response_time - Most Recent Response Time for Connect	470
con_elapsed_time - Most Recent Connection Elapsed Time	470
gw_comm_errors - Communication Errors	471
gw_comm_error_time - Communication Error Time	471
blocking_cursor - Blocking Cursor Transaction processor monitoring	472
Federated database systems	474
Federated database systems monitor elements	474
datasource_name - Data Source Name	475
disconnects - Disconnects	475
insert_sql_stmts - Inserts	475
update_sql_stmts - Updates	476

delete_sql_stmts - Deletes	476
create_nickname - Create Nicknames	477
passthru - Pass-Through	477
stored_procs - Stored Procedures	478
remote_locks - Remote Locks	478
sp_rows_selected - Rows Returned by Stored Procedures	479
select_time - Query Response Time	479
insert_time - Insert Response Time	480
update_time - Update Response Time	480
delete_time - Delete Response Time	481
create_nickname_time - Create Nickname Response Time	481
passthru_time - Pass-Through Time	482
stored_proc_time - Stored Procedure Time	482
remote_lock_time - Remote Lock Time	483

Chapter 7. Monitor Interfaces 485

Database system monitor interfaces	485
Activity Monitor overview	488
Setting up an activity monitor	492
Memory Visualizer overview	492
Working with the Memory Visualizer	495
Indoubt Transaction Manager overview	497

Part 3. Health Monitor Guide 501

Chapter 8. Introducing the health monitor 503

Introduction to the health monitor	503
Health indicators	503
Health indicator process cycle	505
Enabling health alert notification	507

Chapter 9. Using the health monitor 511

Health monitor	511
Health indicator data	512
Capturing a database health snapshot using SQL table functions	513
Health monitor SQL table functions	514
Capturing a database health snapshot using the CLP	515
Health monitor CLP commands	516
Capturing a database health snapshot from a client application	516
Health monitor API request types	519
Health monitor sample output	520
Global health snapshots	522
Graphical tools for the health monitor	523
Health Center overview	525

Part 4. Health Monitor Reference 529

Chapter 10. Health Monitor Logical Data Groups 531

Health monitor interface mappings to logical data groups	531
--	-----

Chapter 11. Health Indicators	533
Health indicator format	533
Health indicators summary	533
Health indicators for DMS table spaces	535
Database storage health indicators	536
db.auto_storage_util – Database automatic storage utilization health indicator	536
Table space storage health indicators	537
ts.ts_auto_resize_status – Table space automatic resize status health indicator	537
ts.ts_util_auto_resize – Automatic resize table space utilization health indicator	538
ts.ts_util - Table Space Utilization	538
tsc.tscont_util - Table Space Container Utilization	539
ts.ts_op_status - Table Space Operational State	540
tsc.tscont_op_status - Table Space Container Operational State	541
Sorting health indicators	541
db2.sort_privmem_util - Private Sort Memory Utilization	541
db.sort_shrmem_util - Shared Sort Memory Utilization	542
db.spilled_sorts - Percentage of Sorts That Overflowed	543
db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization	544
Database manager (DBMS) health indicators	544
db2.db2_op_status - Instance Operational State	544
Instance Highest Severity Alert State	545
Database health indicators	546
db.db_op_status - Database Operational State	546
Database Highest Severity Alert State	546
Maintenance health indicators	546
db.tb_reorg_req - Reorganization Required	546
db.tb_runstats_req - Statistics Collection Required	547
db.db_backup_req - Database Backup Required	548
High availability disaster recovery health indicators	548
db.hadr_op_status - HADR Operational Status	548
db.hadr_delay - HADR Log Delay	549
Logging health indicators	549
db.log_util - Log Utilization	549
db.log_fs_util - Log Filesystem Utilization	550
Application concurrency health indicators	550
db.deadlock_rate - Deadlock Rate	550
db.locklist_util - Lock List Utilization	551
db.lock_escal_rate - Lock Escalation Rate	552

db.apps_waiting_locks - Percentage of Applications Waiting on Locks	553
Package and catalog caches, and workspaces health indicators	553
db.catcache_hitratio - Catalog Cache Hit Ratio	553
db.pkgcache_hitratio - Package Cache Hit Ratio	554
db.shrworkspace_hitratio - Shared Workspace Hit Ratio	554
Memory health indicators	555
db2.mon_heap_util - Monitor Heap Utilization	555
db.db_heap_util - Database Heap Utilization	555
Federated health indicators	556
db.fed_nicknames_op_status - Nickname Status	556
db.fed_servers_op_status - Data Source Server Status	556

Chapter 12. Health Monitor Interfaces	559
Health monitor interfaces	559

Part 5. Appendixes 561

Appendix A. DB2 Database technical information	563
Overview of the DB2 technical information	563
Documentation feedback	563
DB2 technical library in hardcopy or PDF format	564
Ordering printed DB2 books	566
Displaying SQL state help from the command line processor	567
Accessing different versions of the DB2 Information Center	568
Displaying topics in your preferred language in the DB2 Information Center	568
Updating the DB2 Information Center installed on your computer or intranet server	569
DB2 tutorials	571
DB2 troubleshooting information	571
Terms and Conditions	572

Appendix B. Notices	573
Trademarks	575

Index	577
------------------------	------------

Contacting IBM	589
---------------------------------	------------

Part 1. System Monitor Guide

Chapter 1. Introducing the Database System Monitor

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.

Related concepts:

- “Counter status and visibility” on page 7
- “Database system monitor data organization” on page 3
- “Database system monitor memory requirements” on page 9
- “Event monitors” on page 59
- “Snapshot monitor” on page 21

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.

Introduction

Water mark	Indicates the highest (maximum) or lowest (minimum) value an element has reached since monitoring was started. Water mark 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	<p>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 <code>TIMESTAMP</code> 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.</p> <p>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.</p>
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 <code>TIMESTAMP</code> 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.

Related concepts:

- "Counter status and visibility" on page 7
- "Database system monitor" on page 3
- "Event monitors" on page 59
- "Snapshot monitor" on page 21
- "System monitor switches" on page 13

Related reference:

- "RESET MONITOR command" in *Command Reference*

Comparison of DB2 monitors

DB2 Version 9.1 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 9.1 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.
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 update monitor switches command) or at the database manager level (using the update dbm cfg command).	Create event monitor using the <code>AUTOSTART</code> option, or set event monitor to state 1 ²	Enabled by default. To deactivate, set <code>health_mon</code> database manager configuration parameter to OFF

Introduction

Table 1. Comparison of DB2 Version 9.1 Monitors (continued)

	Snapshot monitor	Event monitor	Health monitor
When data is collected	User issues snapshot table functions, snapshot APIs, SNAP_WRITE_FILE stored procedure or get snapshot command from the CLP, or issues SELECT from snapshot administrative views	Specified event occurs ³	Collected by default at preset intervals
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.

Notes:

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.

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.

Related concepts:

- “System monitor switches” on page 13
- “Database system monitor” on page 3
- “Database system monitor data organization” on page 3
- “Event monitors” on page 59
- “Snapshot monitor” on page 21

Related reference:

- “RESET MONITOR command” in *Command Reference*

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 (<i>db</i>) contains individual monitor elements (such as <i>total_log_used</i>) along with other logical data groupings, such as rollforward information (<i>rollforward</i>). 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 <i>header</i> 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, <i>collected</i> , <i>dbase</i> , or <i>event_db</i>).
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.

Related concepts:

- "Event monitor self-describing data stream" on page 91
- "Event type mappings to logical data groups" on page 129
- "Monitor switches self-describing data stream" on page 19
- "Snapshot monitor self-describing data stream" on page 54

Related reference:

- “Snapshot monitor interface mappings to logical data groups” on page 97

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` if monitor commands fail with an SQLCODE of -973.

The following formula provides an approximation of the number of pages required for the monitor heap:

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

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:

- 1300 bytes
- 2 * `BUFFERSIZE`
- If the event monitor is written to a File, add 308 bytes.
- If the event monitor is for type DATABASE:
 - add 2700 bytes
 - add 100 bytes for each statement in the statement cache
- If the event monitor is for type TABLES:
 - add 600 bytes
 - add 75 bytes for each table accessed
- If the event monitor is for type TABLESPACES:
 - add 10 bytes
 - add 250 bytes for each table space
- If the event monitor is for type BUFFERPOOLS:
 - add 10 bytes
 - add 250 bytes for each buffer pool
- If the event monitor is for type CONNECTIONS:

Introduction

- add 600 bytes
- for each connected application:
 - add 600 bytes
- remember to add the "Storage used by applications" above
- If an event monitor is of type DEADLOCK:
 - And the WITH DETAILS HISTORY is running:
 - add $X*100$ 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

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

Related concepts:

- “Database system monitor” on page 3
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from the CLP” on page 15

Related reference:

- “mon_heap_sz - Database system monitor heap size configuration parameter” in *Performance Guide*

Introduction

Chapter 2. System Monitor Switches

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.

System monitor switches

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

Related concepts:

- “Event monitors” on page 59
- “Monitor switches self-describing data stream” on page 19
- “Snapshot monitor” on page 21

Related tasks:

- “Setting monitor switches from a client application” on page 17
- “Setting monitor switches from the CLP” on page 15

Setting monitor switches from the CLP

Before capturing a snapshot or using an event monitor, first determine what data you need the database manager to gather. If you want any of the following special types of data to be collected, set the appropriate monitor switches.

- Buffer pool activity information
- 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.

Note: 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.

Prerequisites:

The application performing any monitor switch updates must have an instance attachment.

You must have one of SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to use the following commands:

- UPDATE MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET DATABASE MANAGER MONITOR SWITCHES

You must have SYSADM authority to use the UPDATE DBM CFG command.

Procedure (UPDATE MONITOR SWITCHES):

- To activate any of the local monitor switches use the 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
```

The switches will remain active until the application (CLP) detaches, or until they are deactivated with another UPDATE MONITOR SWITCHES command.

- 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
```

System monitor switches

```
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.

```
db2 update dbm cfg using DFT_MON_LOCK on
```

In the above example, only lock switch controlled information is to be collected in addition to the basic information.

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. 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
```

To set a monitor switch (for example, `SORT`) for all partitions, issue the following command:

```
db2 update monitor switches using SORT on global
```

Procedure (GET MONITOR SWITCHES):

- 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. 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
```

To view the monitor switch settings for all partitions, issue the following command:

```
db2 get monitor switches global
```

Procedure (GET DATABASE MANAGER MONITOR SWITCHES):

- 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.

Related concepts:

- “Event monitors” on page 59
- “Snapshot monitor” on page 21
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from a client application” on page 17

Setting monitor switches from a client application

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, 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.

Note: 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.

Prerequisites:

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.

Procedure:

1. Include the following DB2 libraries: `sqlutil.h` and `db2ApiDf.h`. These are found in the `include` subdirectory under `sqllib`.

System monitor switches

```
#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_VERSION8;
switchesData.iBufferSize = switchesBufferSize;
switchesData.iReturnData = 0;
switchesData.iNodeNumber = SQLM_CURRENT_NODE;
switchesData.poOutputFormat = &outputFormat;
```

Note that `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.

Related concepts:

- “Event monitors” on page 59

- “Monitor switches self-describing data stream” on page 19
- “Snapshot monitor” on page 21
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from the CLP” on page 15

Related reference:

- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*

Related samples:

- “dbsnap.cbl -- Get a database monitor snapshot (IBM COBOL)”
- “clisnap.out -- HOW TO GET A CLIENT LEVEL SNAPSHOT (C)”
- “dbsnap.out -- HOW TO GET A DATABASE LEVEL SNAPSHOT (C)”
- “insnap.out -- HOW TO GET AN INSTANCE LEVEL SNAPSHOT (C)”
- “clisnap.c -- Capture a snapshot at the client level (C)”
- “dbsnap.c -- Capture a snapshot at the database level (C)”
- “insnap.c -- Capture a snapshot at the instance level (C)”
- “utilsnap.c -- Utilities for the snapshot monitor samples (C)”
- “clisnap.out -- HOW TO GET A CLIENT LEVEL SNAPSHOT (C++)”
- “dbsnap.out -- HOW TO GET A DATABASE LEVEL SNAPSHOT (C++)”
- “insnap.out -- HOW TO GET AN INSTANCE LEVEL SNAPSHOT (C++)”
- “clisnap.C -- Capture a snapshot at the client level (C++)”
- “dbsnap.C -- Capture a snapshot at the database level (C++)”
- “insnap.C -- Capture a snapshot at the instance level (C++)”
- “utilsnap.C -- Utilities for the snapshot monitor samples (C++)”

Monitor switches self-describing data stream

After you update or view the current monitor switch settings with the db2MonitorSwitches API, the API returns the switch settings as a self-describing data stream. Figure 1 on page 20 shows the structure of the switch list information that may be returned for a partitioned database environment.

Notes:

1. Descriptive names are used for the identifiers in the examples and tables. These names are prefixed by **SQLM_ELM_** in the actual data stream. For example, db_event would appear as SQLM_ELM_DB_EVENT in the event monitor output. Types are prefixed with **SQLM_TYPE_** in the actual data stream. For example, headers appear as SQLM_TYPE_HEADER in the data stream.
2. For global switch requests the partition order of the returned information can be different in each switch request. In this case, a partition id is included in the data stream.

System monitor switches

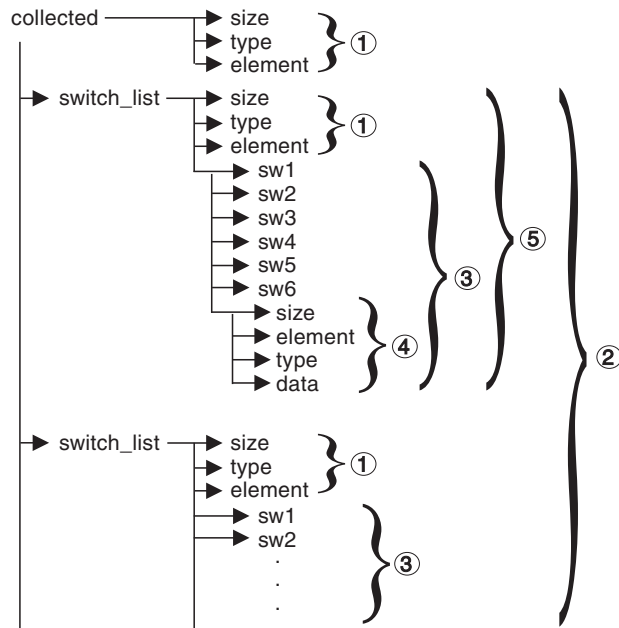


Figure 1. Switch List Monitor Data Stream

1. Each logical data group begins with a header that indicates its size and name. This size does not include the volume of data taken up by the header itself.
2. Size in the collected header returns the total size of all monitor switch lists for all partitions.
3. The size element in switch list header indicates the size of switch data for that partition.
4. Switch information is self-describing.
5. For a non-partitioned database, the switch settings for the stand alone partition are returned. That is, only one switch list is returned.

Related concepts:

- “System monitor output: the self-describing data stream” on page 8
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from a client application” on page 17

Related reference:

- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*

Chapter 3. Using the Snapshot Monitor

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.

Related concepts:

- “Database system monitor” on page 3
- “Database system monitor data organization” on page 3
- “Global snapshots on partitioned database systems” on page 53
- “Subsection snapshots” on page 52
- “System monitor switches” on page 13

Related tasks:

- “Capturing a database snapshot from a client application” on page 46
- “Capturing a database snapshot from the CLP” on page 43
- “SQL access to database system snapshots” on page 57

Snapshot monitor

Related reference:

- “Snapshot monitor sample output” on page 50

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

Related concepts:

- “System monitor authority (SYSMON)” in *Administration Guide: Implementation*
- “Snapshot monitor” on page 21

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.

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
```

Prerequisites:

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.

Restrictions:

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.

Procedure:

To capture a snapshot using a snapshot administrative view you must:

1. Connect to a database. This can be any database in the instance you need to monitor. To be able to issue an SQL query with a snapshot administrative view, you must be connected to a database.
2. Determine the type of snapshot you need to capture. If you want to capture a snapshot for a database other than the currently connected database, or if you want to retrieve data from a single database partition, or global aggregate data, you need to use a snapshot table function instead.

Snapshot monitor

3. Issue a query with the appropriate snapshot administrative view. For example, here is a query that captures a snapshot of lock information for the currently connected database:

```
SELECT * FROM SYSIBMADM.SNAPLOCK
```

To capture a snapshot using a snapshot table function you must:

1. Connect to a database. This can be any database in the instance you need to monitor. To be able to issue an SQL query with a snapshot table function, you must be connected to a database.
2. Determine the type of snapshot you need to capture.
3. Issue a query with the appropriate snapshot table function. For example, here is a query that captures a snapshot of lock information about the SAMPLE database for the current connected database partition:

```
SELECT * FROM TABLE(SNAP_GET_LOCK('SAMPLE',-1)) AS SNAPLOCK
```

The SQL table functions have two input parameters:

database name

VARCHAR(255). If you enter NULL, the name of the currently connected database is used.

partition number

SMALLINT. For the database partition number parameter, enter the integer (a value between 0 and 999) corresponding to the database partition number you need to monitor. To capture a snapshot for the currently connected database partition, enter a value of -1. To capture a global aggregate snapshot, enter a value of -2. To capture a snapshot from all database partitions, do not specify a value for this parameter.

Notes:

- a. For the following list of snapshot table functions, if you enter a NULL for the currently connected database, you will get snapshot information for all databases in the instance:
 - SNAP_GET_DB_V91
 - SNAP_GET_DB_MEMORY_POOL
 - SNAP_GET_DETAILLOG_V91
 - SNAP_GET_HADR
 - SNAP_GET_STORAGE_PATHS
 - SNAP_GET_APPL
 - SNAP_GET_APPL_INFO
 - SNAP_GET_AGENT
 - SNAP_GET_AGENT_MEMORY_POOL
 - SNAP_GET_STMT
 - SNAP_GET_SUBSECTION
 - SNAP_GET_BP
 - SNAP_GET_BP_PART
- b. The database name parameter does not apply to the database manager level snapshot table functions; they have only a parameter for database partition number. The database partition number parameter is optional.

Related concepts:

- “Snapshot monitor” on page 21

- “Administrative SQL routines and views” in *Administrative SQL Routines and Views*

Related reference:

- “Snapshot monitor interface mappings to logical data groups” on page 97
- “Snapshot monitor SQL Administrative Views” on page 37
- “Administrative views versus table functions” in *Administrative SQL Routines and Views*

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: No actual data from tables or user passwords can be exposed using the snapshot monitor table functions.

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	APPL_ALL
SNAP_GET_APPL_INFO	APPL_ALL
SNAP_GET_STMT	APPL_ALL
SNAP_GET_SUBSECTION	APPL_ALL
SNAP_GET_BP_PART	BUFFERPOOLS_ALL
SNAP_GET_BP	BUFFERPOOLS_ALL
SNAP_GET_DB_V91	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	DB2
SNAP_GET_DBM_MEMORY_POOL	DB2

Snapshot monitor

Table 3. Snapshot request types (continued)

Snapshot table function	Snapshot request type
SNAP_GET_FCM	DB2
SNAP_GET_FCM_PART	DB2
SNAP_GET_SWITCHES	DB2
SNAP_GET_DYN_SQL_V91	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

Prerequisites:

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

Procedure:

To capture a snapshot to a file using the SNAP_WRITE_FILE stored procedure:

1. Connect to a database. This can be any database in the instance you need to monitor. To be able to call a stored procedure, you must be connected to a database.
2. Determine the snapshot request type, and the database and partition you need to monitor.
3. Call the SNAP_WRITE_FILE stored procedure with the appropriate parameter settings for the snapshot request type, database, and partition. For example, here is a call that will capture a snapshot of application information about the SAMPLE database for the current connected partition:

```
CALL SNAP_WRITE_FILE('APPL_ALL', 'SAMPLE', -1)
```

The SNAP_WRITE_FILE stored procedure has three input parameters:

- a snapshot request type (see Table 3 on page 25, 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.

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.

Related concepts:

- “Snapshot monitor” on page 21

Related tasks:

- “Accessing database system snapshots using snapshot table functions in SQL queries (with file access)” on page 27

Related reference:

- “Snapshot monitor interface mappings to logical data groups” on page 97
- “Snapshot monitor SQL Administrative Views” on page 37
- “SNAP_WRITE_FILE procedure” in *Administrative SQL Routines and Views*

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.

Users access snapshot data from SNAP_WRITE_FILE files by using snapshot table functions in SQL queries. For example, a snapshot of general application information for the SAMPLE database is made as follows:

Snapshot monitor

```
SELECT * FROM TABLE( SNAP_GET_APPL(CAST(NULL AS VARCHAR(1)),
                                   CAST (NULL AS INTEGER)))
                    as SNAP_GET_APPL
```

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.

You can also select individual monitor elements from the returned table. For example, the following statement will return only the agent_id monitor element:

```
SELECT agent_id FROM TABLE(
                        SNAP_GET_APPL(CAST(NULL AS VARCHAR(1)),
                                       CAST (NULL AS INTEGER)))
                    as SNAP_GET_APPL
```

Prerequisites:

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.

Restrictions:

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.

If you issue an SQL query containing a snapshot table function for which a corresponding SNAP_WRITE_FILE request type has not been run, a direct snapshot is attempted for the currently connected database and partition. This operation is successful only if the user has SYSADM, SYSCTRL, SYSMOINT, or SYSMON authority.

Procedure:

To access snapshot data from SNAP_WRITE_FILE files using a snapshot table function you must:

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.

Related concepts:

- “Snapshot monitor” on page 21

Related tasks:

- “SQL access to database system snapshots” on page 57
- “Capturing database system snapshot information to a file using the SNAP_WRITE_FILE stored procedure” on page 25
- “Capturing database system snapshots using snapshot administrative views and table functions” on page 22

Related reference:

- “Snapshot monitor interface mappings to logical data groups” on page 97
- “Snapshot monitor SQL Administrative Views” on page 37

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

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,

Snapshot monitor

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

```

Snapshot monitor

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/NODE00000/SQL000001/
Input database alias             = DPARTDB
  
```

Snapshot monitor

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:

Snapshot monitor

```
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
```

Related concepts:

- “Data partitions” in *Administration Guide: Planning*
- “Partitioned tables” in *Administration Guide: Planning*
- “Snapshot monitor” on page 21

Related reference:

- “data_partition_id - Data Partition Identifier monitor element” on page 188
- “GET SNAPSHOT command” in *Command Reference*
- “reorg_max_phase - Maximum Reorganize Phase ” on page 365
- “reorg_phase_start - Reorganize Phase Start Time ” on page 365
- “Snapshot monitor interface mappings to logical data groups” on page 97
- “Snapshot monitor logical data groups and monitor elements” on page 100
- “Snapshot monitor sample output” on page 50
- “SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information” in *Administrative SQL Routines and Views*
- “SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function – Retrieve table reorganization snapshot information” in *Administrative SQL Routines and Views*

Snapshot monitor SQL Administrative Views

There are a number of different snapshot monitor SQL administrative views available, each returning monitor data about a specific area of the database system. For example, the SYSIBMADM.SNAPBPB SQL administrative view captures a snapshot of buffer pool information. The following table lists each available snapshot monitor administrative view:

Table 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.
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).

Snapshot monitor

Table 4. Snapshot Monitor SQL Administrative Views (continued)

Monitor level	SQL Administrative Views	Information returned
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 was accessed by an application connected to the database. Requires the table switch.
Table	SYSIBMADM.SNAPTAB_REORG	Table reorganization information at the table level for each table in the database undergoing reorganization.
Lock	SYSIBMADM.SNAPLOCK	Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.
Table space	SYSIBMADM.SNAPTbsp	Information about table space activity at the database level, the application level for each application connected to the database, and the table space level for each table space that has been accessed by an application connected to the database. Requires the buffer pool switch.
Table space	SYSIBMADM.SNAPTbsp_PART	Information about table space configuration.
Table space	SYSIBMADM.SNAPTbsp_QUIESCER	Information about quiescers at the table space level.
Table space	SYSIBMADM.SNAPCONTAINER	Information about table space container configuration at the table space level.
Table space	SYSIBMADM.SNAPTbsp_RANGE	Information about ranges for a table space map.
Buffer pool	SYSIBMADM.SNAPBP	Buffer pool activity counters for the specified database. Requires the buffer pool switch.
Buffer pool	SYSIBMADM.SNAPBP_PART	Information on buffer size and usage, calculated per partition.

Table 4. Snapshot Monitor SQL Administrative Views (continued)

Monitor level	SQL Administrative Views	Information returned
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 9.1 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.
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.

Snapshot monitor

Related concepts:

- “Snapshot monitor” on page 21

Related tasks:

- “SQL access to database system snapshots” on page 57

Related reference:

- “Snapshot monitor interface mappings to logical data groups” on page 97

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;
```

Related concepts:

- “Scenario: Monitoring buffer pool efficiency using administrative views” on page 42

Related reference:

- “APPL_PERFORMANCE administrative view – Retrieve percentage of rows selected for an application” in *Administrative SQL Routines and Views*
- “APPLICATIONS administrative view – Retrieve connected database application information” in *Administrative SQL Routines and Views*
- “LONG_RUNNING_SQL administrative view” in *Administrative SQL Routines and Views*
- “QUERY_PREP_COST administrative view – Retrieve statement prepare time information” in *Administrative SQL Routines and Views*
- “Snapshot monitor SQL Administrative Views” on page 37

- “TOP_DYNAMIC_SQL administrative view – Retrieve information on the top dynamic SQL statements” in *Administrative SQL Routines and Views*

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.

Related concepts:

- “Scenario: Identifying costly applications using snapshot administrative views” on page 40

Related reference:

- “BP_HITRATIO administrative view – Retrieve bufferpool hit ratio information” in *Administrative SQL Routines and Views*

- “BP_READ_IO administrative view – Retrieve bufferpool read performance information” in *Administrative SQL Routines and Views*
- “BP_WRITE_IO administrative view – Retrieve bufferpool write performance information” in *Administrative SQL Routines and Views*
- “Snapshot monitor SQL Administrative Views” on page 37

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 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.

Prerequisites:

You must have SYSADM, SYSCTRL, SYSMANT, or SYSMON authority to capture a database snapshot.

Procedure:

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
```

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.

Related concepts:

- “Global snapshots on partitioned database systems” on page 53
- “Snapshot monitor” on page 21
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from the CLP” on page 15

Related reference:

- “GET SNAPSHOT command” in *Command Reference*
- “Snapshot monitor CLP commands” on page 44
- “Snapshot monitor interface mappings to logical data groups” on page 97

Snapshot monitor

- “Snapshot monitor sample output” on page 50

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 <i>applid 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 <i>agentid appl-handle</i></code>	Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	<code>get snapshot for applications on <i>dbname</i></code>	Application level information for each application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	<code>get snapshot for all applications</code>	Application level information for each application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).

Table 6. Snapshot Monitor CLP Commands (continued)

Monitor level	CLP command	Information returned
Application	get snapshot for dcs application applid <i>appl-id</i>	Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	get snapshot for all dcs applications	Application level information for each DCS application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	get snapshot for dcs application agentid <i>appl-handle</i>	Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	get snapshot for dcs applications on <i>dbname</i>	Application level information for each DCS application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	get snapshot for remote applications on <i>dbname</i>	Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	get snapshot for all remote applications	Application level information for each federated system application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Table	get snapshot for tables on <i>dbname</i>	Table activity information at the database and application level for each application connected to the database. Table activity information at the table level for each table that was accessed by an application connected to the database. Requires the table switch.
Lock	get snapshot for locks for application applid <i>appl-id</i>	List of locks held by the application. Lock wait information requires the lock switch.
Lock	get snapshot for locks for application agentid <i>appl-handle</i>	List of locks held by the application. Lock wait information requires the lock switch.
Lock	get snapshot for locks on <i>dbname</i>	Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.
Table space	get snapshot for tablespaces on <i>dbname</i>	Information about table space activity for a database. Requires the buffer pool switch. Also included is information on containers, quiescers, and ranges. This information is not under switch control.
Buffer pool	get snapshot for all bufferpools	Buffer pool activity counters. Requires the buffer pool switch.
Buffer pool	get snapshot for bufferpools on <i>dbname</i>	Buffer pool activity counters for the specified database. Requires the buffer pool switch.
Dynamic SQL	get snapshot for dynamic sql on <i>dbname</i>	Point-in-time statement information from the SQL statement cache for the database. The information can also be from a remote data source.

Related tasks:

- “Capturing a database snapshot from the CLP” on page 43
- “Setting monitor switches from the CLP” on page 15

Related reference:

- “GET SNAPSHOT command” in *Command Reference*
- “Snapshot monitor interface mappings to logical data groups” on page 97

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 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.

Prerequisites:

You must have `SYSADM`, `SYSCTRL`, `SYSMAINT`, or `SYSMON` authority to use the `db2MonitorSwitches` API.

Procedure:

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.

```
#define SNAPSHOT_BUFFER_UNIT_SZ 102400

struct sqlma *pRequestedDataGroups;
struct sqlca sqlca;
memset (&sqlca, '\0', sizeof(struct sqlca));
struct sqlm_collected collectedData;
memset (&collectedData, '\0', sizeof(collectedData));
db2GetSnapshotData getSnapshotParam;
memset (&getSnapshotParam, '\0', sizeof(getSnapshotParam));

static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
sqluint32 outputFormat;
char *snapshotBuffer;
```

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_VERSION8;
getSnapshotParam.iBufferSize = snapshotBufferSize;
getSnapshotParam.iStoreResult = 0;
```



```
getSnapshotParam.iNodeNumber = SQLM_CURRENT_NODE;
getSnapshotParam.poOutputFormat = &outputFormat;
getSnapshotParam.iSnapshotClass = SQLM_CLASS_DEFAULT;
```

- Capture the snapshot. Pass the db2GetSnapshotData structure, which contains the information necessary to capture a snapshot, as well as a reference to the buffer, where snapshot output is to be directed.

```
db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
```

- Include logic to handle buffer overflow. After a snapshot is taken, the sqlcode is checked for a buffer overflow. If a buffer overflow occurred the buffer is cleared and reinitialized, and the snapshot is taken again.

```
while (sqlca.sqlcode == 1606)
{
    free(snapshotBuffer);
    snapshotBufferSize = snapshotBufferSize +
    SNAPSHOT_BUFFER_UNIT_SZ;
    snapshotBuffer = (char *)malloc(snapshotBufferSize);
    if (snapshotBuffer == NULL)
    {
        printf("\nMemory allocation error.\n");
        return 1;
    }
    getSnapshotParam.iBufferSize = snapshotBufferSize;
    getSnapshotParam.poBuffer = snapshotBuffer;
    db2GetSnapshot(db2Version810, &getSnapshotParam, &sqlca);
}
```

- Process the snapshot monitor data stream.

- Clear the buffer.

```
free(snapshotBuffer);
free(pRequestedDataGroups);
```

Related concepts:

- “Snapshot monitor” on page 21
- “Snapshot monitor self-describing data stream” on page 54
- “System monitor output: the self-describing data stream” on page 8
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from a client application” on page 17

Related reference:

- “db2GetSnapshot API - Get a snapshot of the database manager operational status” in *Administrative API Reference*
- “db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API” in *Administrative API Reference*
- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*
- “db2ResetMonitor API - Reset the database system monitor data” in *Administrative API Reference*
- “Snapshot monitor API request types” on page 48

Related samples:

- “dbsnap.cbl -- Get a database monitor snapshot (IBM COBOL)”
- “clisnap.out -- HOW TO GET A CLIENT LEVEL SNAPSHOT (C)”
- “dbsnap.out -- HOW TO GET A DATABASE LEVEL SNAPSHOT (C)”

Snapshot monitor

- “insnap.out -- HOW TO GET AN INSTANCE LEVEL SNAPSHOT (C)”
- “clisnap.c -- Capture a snapshot at the client level (C)”
- “dbsnap.c -- Capture a snapshot at the database level (C)”
- “insnap.c -- Capture a snapshot at the instance level (C)”
- “utilsnap.c -- Utilities for the snapshot monitor samples (C)”
- “clisnap.out -- HOW TO GET A CLIENT LEVEL SNAPSHOT (C++)”
- “dbsnap.out -- HOW TO GET A DATABASE LEVEL SNAPSHOT (C++)”
- “insnap.out -- HOW TO GET AN INSTANCE LEVEL SNAPSHOT (C++)”
- “clisnap.C -- Capture a snapshot at the client level (C++)”
- “dbsnap.C -- Capture a snapshot at the database level (C++)”
- “insnap.C -- Capture a snapshot at the instance level (C++)”
- “utilsnap.C -- Utilities for the snapshot monitor samples (C++)”

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.
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.

Table 7. Snapshot Monitor API Request Types (continued)

Monitor level	API request type	Information returned
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).
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.

Snapshot monitor

Table 7. Snapshot Monitor API Request Types (continued)

Monitor level	API request type	Information returned
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.

Related concepts:

- “Snapshot monitor” on page 21
- “Snapshot monitor self-describing data stream” on page 54

Related tasks:

- “Capturing a database snapshot from a client application” on page 46
- “Setting monitor switches from a client application” on page 17

Related reference:

- “db2ConvMonStream API - Convert the monitor stream to the pre-version 6 format” in *Administrative API Reference*
- “db2GetSnapshot API - Get a snapshot of the database manager operational status” in *Administrative API Reference*
- “db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API” in *Administrative API Reference*
- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*
- “db2ResetMonitor API - Reset the database system monitor data” in *Administrative API Reference*

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
```

Snapshot monitor

```
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
```

Related concepts:

- “Snapshot monitor” on page 21
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from the CLP” on page 15

Related reference:

- “Snapshot monitor CLP commands” on page 44

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.

Related concepts:

- “Global snapshots on partitioned database systems” on page 53
- “Snapshot monitor” on page 21
- “System monitor switches” on page 13

Related reference:

- “Snapshot monitor CLP commands” on page 44
- “GET SNAPSHOT command” in *Command Reference*

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.
- **Water marks**
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**

Snapshot monitor

Set to the timestamp value for the partition where the snapshot monitor instance agent is attached. Note that all timestamp values are under control of the timestamp monitor switch.

- **Information**

Returns the most significant information for a partition that may be impeding work. For example, for the element `appl_status`, if the status on one partition was UOW Executing, and on another partition Lock Wait, Lock Wait would be returned, since it is the state that's holding up execution of the application.

You can also reset counters, set monitor switches, and retrieve monitor switch settings for individual partitions or all partitions in your partitioned database.

Note: When taking a global snapshot, if one or more partitions encounter an error, then data is collected from the partitions where the snapshot was successful and a warning (sqlcode 1629) is also returned. If a global get or update of monitor switches, or a counter reset fails on one or more partitions, then those partitions will not have their switches set, or data reset.

Related concepts:

- “Counter status and visibility” on page 7
- “Snapshot monitor” on page 21
- “Subsection snapshots” on page 52

Related tasks:

- “Capturing a database snapshot from a client application” on page 46
- “Capturing a database snapshot from the CLP” on page 43
- “SQL access to database system snapshots” on page 57

Snapshot monitor self-describing data stream

After you capture a snapshot with the `db2GetSnapshot` API, the API returns the snapshot output as a self-describing data stream. Figure 2 on page 55 shows the structure of the data stream and Table 8 on page 56 provides some examples of the logical data groups and monitor elements that might be returned.

Note: Descriptive names are used for the identifiers in the examples and tables. These names are prefixed by `SQLM_ELM_` in the actual data stream. For example, collected would appear as `SQLM_ELM_COLLECTED` in the snapshot monitor output. Types are prefixed with `SQLM_TYPE_` in the actual data stream. For example, headers appear as `SQLM_TYPE_HEADER` in the data stream.

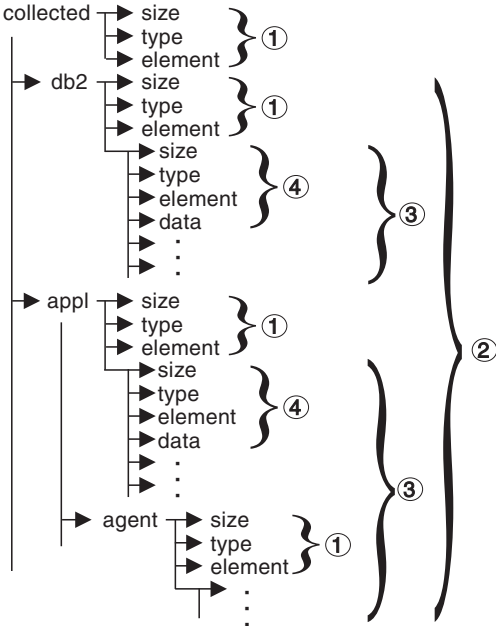


Figure 2. Snapshot Monitor Data Stream

- 1. Each logical data group begins with a header that indicates its size and name. This size does not include the volume of data taken up by the header itself.
- 2. Size in the collected header returns the total size of the snapshot.
- 3. The size element in other headers indicates the size of all the data in that logical data group, including any subordinate groupings.
- 4. Monitor element information follows its logical data group header and is also self-describing.

Snapshot monitor

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.
db2	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.
	200	Size of the DB2 level portion of data in the snapshot.
	header	Indicates the start of a logical data group.
db2	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.
appl	u32bit	Monitor element type - unsigned 32 bit numeric.
	local_cons	The name of the monitor element collected.
	3	The collected value for this element.

	100	Size of the appl element data in the snapshot.
	header	Indicates the start of a logical data group.
appl	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.

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.
agent	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:

size of the db2 logical data grouping + sizeof(sqlm_header_info)

Related concepts:

- "Snapshot monitor" on page 21

Related tasks:

- “Capturing a database snapshot from a client application” on page 46

Related reference:

- “Snapshot monitor API request types” on page 48
- “Snapshot monitor interface mappings to logical data groups” on page 97

Related samples:

- “clisnap.out -- HOW TO GET A CLIENT LEVEL SNAPSHOT (C)”
- “dbsnap.out -- HOW TO GET A DATABASE LEVEL SNAPSHOT (C)”
- “insnap.out -- HOW TO GET AN INSTANCE LEVEL SNAPSHOT (C)”
- “clisnap.c -- Capture a snapshot at the client level (C)”
- “dbsnap.c -- Capture a snapshot at the database level (C)”
- “insnap.c -- Capture a snapshot at the instance level (C)”
- “utilsnap.c -- Utilities for the snapshot monitor samples (C)”
- “clisnap.out -- HOW TO GET A CLIENT LEVEL SNAPSHOT (C++)”
- “dbsnap.out -- HOW TO GET A DATABASE LEVEL SNAPSHOT (C++)”
- “insnap.out -- HOW TO GET AN INSTANCE LEVEL SNAPSHOT (C++)”
- “clisnap.C -- Capture a snapshot at the client level (C++)”
- “dbsnap.C -- Capture a snapshot at the database level (C++)”
- “insnap.C -- Capture a snapshot at the instance level (C++)”
- “utilsnap.C -- Utilities for the snapshot monitor samples (C++)”

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, SYSMAINT, 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

Snapshot monitor

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.

Related concepts:

- “Snapshot monitor” on page 21
- “System monitor switches” on page 13

Related tasks:

- “Setting monitor switches from the CLP” on page 15
- “Capturing database system snapshots using snapshot administrative views and table functions” on page 22
- “Capturing database system snapshot information to a file using the `SNAP_WRITE_FILE` stored procedure” on page 25
- “Accessing database system snapshots using snapshot table functions in SQL queries (with file access)” on page 27

Related reference:

- “Snapshot monitor interface mappings to logical data groups” on page 97
- “Snapshot monitor SQL Administrative Views” on page 37

Chapter 4. Using Event Monitors

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.

Related concepts:

- "Database system monitor" on page 3

Related tasks:

- "Collecting information about database system events" on page 61
- "Creating an event monitor" on page 63

Related reference:

Event monitors

- “Event monitor sample output” on page 80
- “Event types” on page 60

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.

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.

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.

Related concepts:

- “Counter status and visibility” on page 7
- “Event monitors” on page 59
- “Event type mappings to logical data groups” on page 129

Related tasks:

- “Collecting information about database system events” on page 61
- “Creating an event monitor” on page 63

Related reference:

- “DROP statement” in *SQL Reference, Volume 2*
- “Event monitor sample output” on page 80

Collecting information about database system events

Event monitors are database objects, and as such, they are created and manipulated using SQL data definition language (SQL DDL) statements. 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.

1. The first step will always be the creation of the event monitor.
2. The last step will always be the deletion of the event monitor.

Prerequisites:

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

Procedure:

1. Create an event monitor. Create 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.

Event monitors

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[®] 2000, you can create named pipes by using the `CreateNamedPipe()` routine.

3. *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 2000: `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.

4. 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.

5. 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.

6. 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`.
7. 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.

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.

After you drop a pipe event monitor, delete the corresponding named pipe. In UNIX use the `unlink()` function, and in Windows 2000 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.

Related concepts:

- “Event monitors” on page 59
- “Event records and their corresponding applications” on page 90

Related tasks:

- “Creating an event monitor” on page 63
- “Creating an event monitor for partitioned databases” on page 77
- “Formatting file or pipe event monitor output from a command line” on page 79

Related reference:

- “Event monitor sample output” on page 80

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. 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.

Prerequisites:

You will need `DBADM` authority to create an event monitor.

Procedure:

- Create a table event monitor
- Create a file event monitor
- Create a pipe event monitor
- 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.

Related concepts:

- “Event monitor file management” on page 73

Event monitors

- “Event monitor named pipe management” on page 76
- “Event monitor table management” on page 67
- “Event monitors” on page 59

Related tasks:

- “Creating a table event monitor” on page 64
- “Creating a file event monitor” on page 71
- “Creating a pipe event monitor” on page 75
- “Creating an event monitor for partitioned databases” on page 77

Related reference:

- “CREATE EVENT MONITOR statement” in *SQL Reference, Volume 2*
- “Event monitor sample output” on page 80
- “Event types” on page 60

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.

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.

Restriction:

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

Prerequisites:

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

Procedure:

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

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

d1mon 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 d1mon 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.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 sums to 32K of buffer space; 16K for each buffer.

The default (and minimum) value for BUFFERSIZE is 4 pages. For performance reasons, highly active event monitors should have larger buffers than relatively inactive event monitors.

4. Indicate if you need the event monitor to be blocked or non-blocked. For blocked event monitors, each agent that generates an event will wait for the event buffers to be written to table if they are full. This can degrade database performance, as the suspended agent and any dependent agents cannot run until the buffers are clear. Use the BLOCKED clause to ensure no losses of event data:

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

Event monitors are blocked by default.

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

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

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

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

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

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

Event monitors

```
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, table_name are the only monitor elements captured.

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

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

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

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO TABLE DLCONN PCTDEACTIVATE 90
    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.

9. 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 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 dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO TABLE BUFFERSIZE 8 NONBLOCKED
    MANUALSTART
```

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.

Related concepts:

- “Event monitor table management” on page 67
- “Event monitors” on page 59
- “Event records and their corresponding applications” on page 90

Related tasks:

- “Collecting information about database system events” on page 61

Related reference:

- “Event types” on page 60
- “CREATE EVENT MONITOR statement” in *SQL Reference, Volume 2*
- “db2evtbl - Generate event monitor target table definitions command” in *Command Reference*

Event monitor table management

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

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

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

```
CREATE EVENT MONITOR foo FOR STATEMENTS WRITE TO TABLE
```

Event monitors using the STATEMENTS event type collect data from the event_connheader, event_stmt, and event_subsection logical data groups. The database created the following tables:

- riihi.connheader_foo
- riihi.stmt_foo
- riihi.subsection_foo
- riihi.control_foo

In addition to the tables representing logical data groups specific to individual event types, a control table is created for every write-to-table event monitor. This is represented above by riihi.control_foo. A control table contains event monitor metadata, specifically, from the event_start, event_db_header (conn_time monitor element only), and event_overflow logical data groups.

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

Write-to-table event monitor target tables must be pruned manually. On highly active systems, event monitors can quickly fill machine space due to the high volume of data they record. Unlike event monitors that write to files or named pipes, you can define write-to-table event monitors to record only certain logical

Event monitors

data groups, or monitor elements. This feature enables you to collect only the data relevant to your purposes and reduce the volume of data generated by the event monitors. For example, the following statement defines an event monitor that captures TRANSACTIONS events, but only from the event_xact logical data group, and including only the lock_escal monitor element:

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

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

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

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

The table space name can be added after the table name with the optional IN clause. Unlike the target tables, which DB2 automatically creates, a table space must already exist if it is included in an event monitor definition. If no table space is specified, then a table space over which the definer has USE privileges will be assigned.

In a partitioned database environment, a write-to-table event monitor will only be active on database partitions where the table space containing the event monitor table exists. When the target table space for an active event monitor does not exist on a particular database partition, the event monitor will be deactivated on that database partition, and an error is written to the db2diag.log file.

For increased performance in retrieving event monitor data, you can create indexes for the event tables. You can also add additional table attributes, such as triggers, relational integrity, and constraints. The event monitor will ignore them.

For example, the following statement defines an event monitor that captures STATEMENTS events, using the event_connheader, event_stmt, and event_subsection logical data groups. Each of the three target tables has different schema, table and table space combinations:

```
CREATE EVENT MONITOR foo FOR STATEMENTS
WRITE TO TABLE CONNHEADER,
STMT (TABLE mydept.statements),
SUBSECTION (TABLE subsections, IN mytablespace)
```

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

- CONNHEADER: riihi.connheader_foo (on the default table space)
- STMT: mydept.statements (on the default table space)
- SUBSECTION: riihi.subsections (on the MYTABLESPACE table space)

If a target table does not exist when the event monitor activates, activation continues and data that would otherwise be inserted into the target table is ignored. Correspondingly, if a monitor element does not have a dedicated column in the target table, it is ignored.

For active write-to-table event monitors there is a risk that the table spaces storing event records can reach their capacity. To control this risk for DMS table spaces you can define at which percentage of table space capacity the event monitor will deactivate. This can be declared in the PCTDEACTIVATE clause in the CREATE EVENT MONITOR statement.

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

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

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

Event type	Target table names	Available information
DEADLOCKS	CONNHEADER DEADLOCK DLCONN CONTROL	Connection metadata Deadlock data Applications and locks involved in deadlock Event monitor metadata
DEADLOCKS WITH DETAILS	CONNHEADER DEADLOCK DLCONN DLLOCK CONTROL	Connection metadata Deadlock data Applications involved in deadlock Locks involved in deadlock Event monitor metadata
DEADLOCKS WITH DETAILS HISTORY	CONNHEADER DEADLOCK DLCONN DLLOCK STMTHIST CONTROL	Connection metadata Deadlock data Applications involved in deadlock Locks involved in deadlock List of the previous statements in the unit of work Event monitor metadata
DEADLOCKS WITH DETAILS HISTORY DATA VALUES	CONNHEADER DEADLOCK DLCONN DLLOCK STMTHIST STMTHIST CONTROL	Connection metadata Deadlock data Applications involved in deadlock Locks involved in deadlock List of the previous statements in the unit of work Input Data values of statements in STMTHIST table Event monitor metadata
STATEMENTS	CONNHEADER STMT SUBSECTION CONTROL	Connection metadata Statement data Statement data specific to subsection Event monitor metadata
TRANSACTIONS	CONNHEADER XACT CONTROL	Connection metadata Transaction data Event monitor metadata
CONNECTIONS	CONNHEADER CONN CONTROL CONMEMUSE	Connection metadata Connection data Event monitor metadata Memory pool metadata
DATABASE	DB CONTROL DBMEMUSE	Database manager data Event monitor metadata Memory pool metadata
BUFFERPOOLS	BUFFERPOOL CONTROL	Buffer pool data Event monitor metadata
TABLESPACES	TABLESPACE CONTROL	Tablespace data Event monitor metadata
TABLES	TABLE CONTROL	Table data Event monitor metadata

Event monitors

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]

Notes:

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.

Related concepts:

- “Event monitors” on page 59
- “Event type mappings to logical data groups” on page 129
- “Write-to-table and file event monitor buffering” on page 74

Related tasks:

- “Creating a table event monitor” on page 64

Related reference:

- “event_monitor_name - Event Monitor Name ” on page 423

Creating a file event monitor

When creating an event monitor you must determine where the information it collects is to be stored. 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.

File event monitors, and their options are defined by the CREATE EVENT MONITOR statement.

Prerequisites:

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

Procedure:

1. Indicate that event monitor data is to be collected in a file (or set of files), and provide a directory location where event files are to be stored.

```
CREATE EVENT MONITOR d1mon FOR eventtype
      WRITE TO FILE '/tmp/d1events'
```

d1mon is the name of the event monitor.

/tmp/d1events is the name of the directory path (on UNIX) 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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
      WRITE TO FILE '/tmp/d1events'
```

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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
      WRITE TO FILE '/tmp/d1events' BUFFERSIZE 8
```

8 is the capacity in 4K pages of the two event file buffers.

The default (and minimum) value for BUFFERSIZE is 4). 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 d1mon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
      WRITE TO FILE '/tmp/d1events' BUFFERSIZE 8
      BLOCKED
```

Event monitors are blocked by default.

Event monitors

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

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
    NONBLOCKED
```

5. Specify the maximum number of event files that can be collected for an event monitor. If this limit is reached, the event monitor will deactivate itself.

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
    NONBLOCKED MAXFILES 5
```

5 is the maximum number of event files that will be created.

You can also specify that there is no limit to the number of event files that the event monitor can create:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
    NONBLOCKED MAXFILES NONE
```

6. Specify the maximum size (in 4K pages) for each event file created by an event monitor. If this limit is reached, a new file is created.

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
    NONBLOCKED MAXFILES 5 MAXFILESIZE 32
```

32 is the maximum number of 4K pages that an event file can contain.

This value must be greater than the value specified by the `BUFFERSIZE` parameter. You can also specify that there is to be no limit on an event file's size:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
    NONBLOCKED MAXFILES NONE MAXFILESIZE NONE
```

7. Specify if the event monitor is to be activated automatically each time the database starts. By default, event monitors are not activated automatically when the database starts.

- To create an event monitor that starts automatically when the database starts, issue the following statement:

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
    NONBLOCKED AUTOSTART
```

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

```
CREATE EVENT MONITOR dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
    WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
    NONBLOCKED MANUALSTART
```

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.

Related concepts:

- “Event monitor file management” on page 73

- “Event monitors” on page 59
- “Event type mappings to logical data groups” on page 129
- “Write-to-table and file event monitor buffering” on page 74

Related tasks:

- “Collecting information about database system events” on page 61
- “Creating an event monitor for partitioned databases” on page 77
- “Formatting file or pipe event monitor output from a command line” on page 79

Related reference:

- “Event monitor sample output” on page 80
- “Event types” on page 60

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.

Event monitors

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.

Related concepts:

- “Event monitor self-describing data stream” on page 91
- “Event monitors” on page 59
- “Event type mappings to logical data groups” on page 129
- “System monitor output: the self-describing data stream” on page 8
- “Write-to-table and file event monitor buffering” on page 74

Related tasks:

- “Collecting information about database system events” on page 61
- “Creating a file event monitor” on page 71

Related reference:

- “Event monitor sample output” on page 80

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.

Related concepts:

- “Event monitor file management” on page 73
- “Event monitor table management” on page 67
- “Event monitors” on page 59

Related tasks:

- “Creating a file event monitor” on page 71
- “Creating a table event monitor” on page 64

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. 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.

Prerequisites:

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

Procedure:

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.

Event monitors

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
```

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.

Related concepts:

- “Event monitor named pipe management” on page 76
- “Event monitor self-describing data stream” on page 91
- “System monitor output: the self-describing data stream” on page 8

Related reference:

- “Event monitor sample output” on page 80

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.

Related concepts:

- “Event monitor self-describing data stream” on page 91
- “Event monitors” on page 59
- “Event type mappings to logical data groups” on page 129
- “System monitor output: the self-describing data stream” on page 8

Related tasks:

- “Collecting information about database system events” on page 61
- “Creating an event monitor” on page 63

Related reference:

- “Event monitor sample output” on page 80

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. 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

Event monitors

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, that partition is activated before the event monitor is activated. In this case, database activation behaves as if an `SQL CONNECT` statement has activated the database on all partitions.

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.

Prerequisites:

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

Procedure:

1. Specify the partition to be monitored.

```
CREATE EVENT MONITOR d1mon FOR DEADLOCKS
WRITE TO FILE '/tmp/d1events'
ON PARTITION 3
```


d1mon represents the name of the event monitor.

/tmp/d1events 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.

- 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 d1mon FOR DEADLOCKS
    WRITE TO FILE '/tmp/d1events'
    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.

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

```
CREATE EVENT MONITOR d1mon FOR DEADLOCKS
    WRITE TO FILE '/tmp/d1events'
    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.

- 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.

Related concepts:

- “Counter status and visibility” on page 7
- “Event monitors” on page 59
- “Event type mappings to logical data groups” on page 129

Related tasks:

- “Creating a file event monitor” on page 71
- “Creating a table event monitor” on page 64

Related reference:

- “Event monitor sample output” on page 80
- “Event types” on page 60

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).

Event monitors

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.

Prerequisites:

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

- SYSADM
- SYSCTRL
- SYSMANT
- DBADM

Procedure:

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 'dlmon'`

`sample` represents the database the event monitor belongs to.

`dlmon` represents an event monitor.

Related concepts:

- “Event monitor file management” on page 73
- “Event monitor named pipe management” on page 76
- “Event monitor self-describing data stream” on page 91

Related tasks:

- “Creating a file event monitor” on page 71
- “Creating a pipe event monitor” on page 75

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.

Monitor Session

```
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 "insert into staff values(26, 'Simpson',
    2, 'Mgr', 13, 35000, 0)"
```

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

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

Application 2

```
db2 connect to sample
db2 +c "insert into department values('7G', 'Safety',
    '1', 'A00', NULL)"
```

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

Application 1

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

Assuming cursor stability, Application 1 needs a share lock on each row of the department table as the rows are fetched, but a lock on the last row 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 last row of the staff table.

Event monitors

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
. . .
SOFTWARE SUPPORT
9 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:\d1mon
```

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.

Only event monitors that return a version of `SQLM_DBMON_VERSION6`, `SQLM_DBMON_VERSION7`, or `SQLM_DBMON_VERSION8` use the self-describing data stream. Pre-Version 6 output must be read using the Version 5 method. For information on these static sized structures refer to the `sqlmon.h` file.

Prolog:

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

```
-----
                                EVENT LOG HEADER
Event Monitor name: DLMON
Server Product ID: SQL08020
Version of event monitor data: 7
Byte order: LITTLE ENDIAN
Number of nodes in db2 instance: 1
Codepage of database: 1252
Territory code of database: 1
Server instance name: DB2
-----

-----
Database Name: SAMPLE
Database Path: D:\DB2\NODE0000\SQL00001\
First connection timestamp: 11/25/2003 13:07:32.649113
Event Monitor Start time: 11/25/2003 13:07:52.292867
-----
```

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) Connection Header Event ...
  Appl Handle: 12
  Appl Id: *LOCAL.DB2.0063C5180732
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.DB2.0063C5180732
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.DB2.0063C5180732
  Program Name      : db2bp.exe
  Authorization Id: RIIHI
  Execution Id      : RIIHI
  Codepage Id: 1252
  Territory code: 1
  Client Process Id: 312
  Client Database Alias: SAMPLE
  Client Product Id: SQL08020
  Client Platform: Unknown
  Client Communication Protocol: Local
  Client Network Name:
  Connect timestamp: 11/25/2003 13:07:32.649113
```

```
4) Connection Header Event ...
```

Event monitors

App1 Handle: 13
App1 Id: *LOCAL.DB2.00FE05180800
App1 Seq number: 0001
DRDA AS Correlation Token: *LOCAL.DB2.00FE05180800
Program Name : db2bp.exe
Authorization Id: RIIHI
Execution Id : RIIHI
Codepage Id: 1252
Execution Id : RIIHI
Codepage Id: 1252
Territory code: 0
Client Process Id: 2144
Client Database Alias: SAMPLE
Client Product Id: SQL08020
Client Platform: Unknown
Client Communication Protocol: Local
Client Network Name:
Connect timestamp: 11/25/2003 13:08:00.897979

5) Deadlock Event ...

Deadlock ID: 1
Number of applications deadlocked: 2
Deadlock detection time: 11/25/2003 13:09:02.831833
Rolled back App1 participant no: 2
Rolled back App1 Id: *LOCAL.DB2.00FE05180800
Rolled back App1 seq number: : 0001

6) Deadlocked Connection ...

Deadlock ID: 1
Participant no.: 2
Participant no. holding the lock: 1
App1 Id: *LOCAL.DB2.00FE05180800
Participant no. holding the lock: 1
App1 Id: *LOCAL.DB2.00FE05180800
App1 Seq number: 0001
App1 Id of connection holding the lock: *LOCAL.DB2.00FE05180800
Seq. no. of connection holding the lock: 0001
Lock wait start time:
Lock Name : 0x02000300280000000000000052
Lock Attributes : 0x00000000
Release Flags : 0x00000001
Lock Count : 1
Hold Count : 0
Current Mode : none
Deadlock detection time: 11/25/2003 13:09:02.832048
Table of lock waited on : STAFF
Schema of lock waited on : RIIHI

Tablespace of lock waited on : USERSPACE1
Type of lock: Row
Mode of lock: X - Exclusive
Mode application requested on lock: NS - Share (and Next Key Share)
Node lock occurred on: 0
Lock object name: 40
Application Handle: 13
Deadlocked Statement:
Type : Dynamic
Operation: Fetch
Section : 201
Creator : NULLID
Package : SQLC2E03
Cursor : SQLCUR201
Cursor was blocking: FALSE
Text : select name from staff
List of Locks:
Lock Name : 0x010000000100000001004C0056
Lock Attributes : 0x00000000

```

Release Flags          : 0x40000000
Lock Count            : 1
Hold Count            : 0
Lock Object Name      : 0
Object Type           : Internal - Variation
Mode                  : S - Share

Lock Name             : 0x020004000D0000000000000052
Lock Attributes       : 0x00000000
Release Flags        : 0x40000000
Lock Count           : 1
Hold Count           : 0
Lock Count           : 1
Hold Count           : 0
Lock Object Name     : 13
Object Type          : Row
Tablespace Name      : USERSPACE1
Table Schema         : RIIHI
Table Name           : DEPARTMENT
Mode                 : X - Exclusive

Lock Name             : 0x94928D848F9F949E7B89505241
Lock Attributes       : 0x00000000
Release Flags        : 0x40000000
Lock Count           : 1
Hold Count           : 0
Lock Object Name     : 0
Object Type          : Internal - Plan
Mode                 : S - Share

Lock Name             : 0x96A09A989DA09A7D8E8A6C7441
Lock Attributes       : 0x00000000
Release Flags        : 0x40000000
Lock Count           : 1
Hold Count           : 0
Lock Object Name     : 0
Hold Count           : 0
Lock Object Name     : 0
Object Type          : Internal - Plan
Mode                 : S - Share

Lock Name             : 0x020004000000000000000000054
Lock Attributes       : 0x00000000
Release Flags        : 0x40000000
Lock Count           : 1
Hold Count           : 0
Lock Object Name     : 4
Object Type          : Table
Tablespace Name      : USERSPACE1
Table Schema         : RIIHI
Table Name           : DEPARTMENT
Mode                 : IX - Intent Exclusive

Lock Name             : 0x020003000000000000000000054
Lock Attributes       : 0x00000000
Release Flags        : 0x00000001
Lock Count           : 1
Hold Count           : 0
Lock Object Name     : 3
Object Type          : Table
Lock Object Name     : 3
Object Type          : Table
Tablespace Name      : USERSPACE1
Table Schema         : RIIHI
Table Name           : STAFF
Mode                 : IS - Intent Share

```

Event monitors

Locks Held: 6
Locks in List: 6

7) Deadlocked Connection ...

Deadlock ID: 1
Participant no.: 1
Participant no. holding the lock: 2
Appl Id: *LOCAL.DB2.0063C5180732
Appl Seq number: 0002
Appl Id of connection holding the lock: *LOCAL.DB2.0063C5180732
Seq. no. of connection holding the lock: 0002
Lock wait start time:
Lock Name : 0x020004000D0000000000000052
Lock Attributes : 0x00000000
Release Flags : 0x00000001
Lock Count : 1
Hold Count : 0
Lock Count : 1
Hold Count : 0
Current Mode : none
Deadlock detection time: 11/25/2003 13:09:02.968324
Table of lock waited on : DEPARTMENT
Schema of lock waited on : RIIHI
Tablespace of lock waited on : USERSPACE1
Type of lock: Row
Mode of lock: X - Exclusive
Mode application requested on lock: NS - Share (and Next Key Share)
Node lock occurred on: 0
Lock object name: 13
Application Handle: 12
Deadlocked Statement:
Type : Dynamic
Operation: Fetch
Section : 201
Creator : NULLID
Package : SQLC2E03
Cursor : SQLCUR201
Cursor was blocking: FALSE
Text : select deptname from department

List of Locks:

Lock Name	: 0x020004000D0000000000000052
Lock Attributes	: 0x00000000
Release Flags	: 0x00000001
Lock Count	: 1
Hold Count	: 0
Lock Object Name	: 13
Object Type	: Row
Tablespace Name	: USERSPACE1
Table Schema	: RIIHI
Table Name	: DEPARTMENT
Mode	: NS - Share (and Next Key Share)
Lock Name	: 0x01000000010000000100640056
Lock Attributes	: 0x00000000
Release Flags	: 0x40000000
Lock Count	: 1
Hold Count	: 0
Lock Object Name	: 0
Object Type	: Internal - Variation
Mode	: S - Share
Lock Name	: 0x02000300280000000000000052
Lock Attributes	: 0x00000000
Lock Name	: 0x02000300280000000000000052
Lock Attributes	: 0x00000000
Release Flags	: 0x40000000
Lock Count	: 1


```

Hold Count          : 0
Lock Object Name    : 40
Object Type         : Row
Tablespace Name     : USERSPACE1
Table Schema        : RIIHI
Table Name          : STAFF
Mode                : X - Exclusive

Lock Name           : 0x94928D848F9F949E7B89505241
Lock Attributes     : 0x00000000
Release Flags       : 0x40000000
Lock Count          : 1
Hold Count          : 0
Lock Object Name    : 0
Object Type         : Internal - Plan
Mode                : S - Share

Lock Name           : 0x02000400000000000000000000000054
Lock Attributes     : 0x00000000
Release Flags       : 0x00000001
Lock Attributes     : 0x00000000
Release Flags       : 0x00000001
Lock Count          : 1
Hold Count          : 0
Lock Object Name    : 4
Object Type         : Table
Tablespace Name     : USERSPACE1
Table Schema        : RIIHI
Table Name          : DEPARTMENT
Mode                : IS - Intent Share

Lock Name           : 0x02000300000000000000000000000054
Lock Attributes     : 0x00000000
Release Flags       : 0x40000000
Lock Count          : 1
Hold Count          : 0
Lock Object Name    : 3
Object Type         : Table
Tablespace Name     : USERSPACE1
Table Schema        : RIIHI
Table Name          : STAFF
Mode                : IX - Intent Exclusive

```

```

Locks Held: 6
Locks in List: 6

```

Epilog:

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

8) Table Event ...

```

Table schema: SYSIBM
Table name: SYSBOOT

```

```

Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 1
Rows read: 1
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101214

```

9) Table Event ...

```

Table schema: SYSIBM

```

Event monitors

Table name: SYSTABLES
Record is the result of a flush: FALSE
Table type: Catalog
Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 28
Index object pages: 17
Lob object pages: 256
Rows read: 2
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101265

10) Table Event ...
Table schema: SYSIBM
Table name: SYSPLAN
Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 9
Index object pages: 5
Lob object pages: 320
Rows read: 1
Rows written: 0
Overflow Accesses: 0
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101303

11) Table Event ...
Table schema: SYSIBM
Table name: SYSDBAUTH
Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 1
Index object pages: 3
Rows read: 3
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101337

12) Table Event ...
Table schema: SYSIBM
Table name: SYSEVENTMONITORS
Table schema: SYSIBM
Table name: SYSEVENTMONITORS
Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 1
Index object pages: 3
Lob object pages: 64
Rows read: 3
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101382

- 13) Table Event ...
Table schema: SYSIBM
Table name: SYSTABLESPACES
- Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 1
Index object pages: 7
Rows read: 3
Index object pages: 7
Rows read: 3
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101412
- 14) Table Event ...
Table schema: SYSIBM
Table name: SYSBUFFERPOOLS
- Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 1
Index object pages: 4
Rows read: 1
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101452
- 15) Table Event ...
Table schema: SYSIBM
Table name: SYSVERSIONS
- Record is the result of a flush: FALSE
Table type: Catalog
Data object pages: 1
Index object pages: 3
Rows read: 1
Rows written: 0
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 0
Table event timestamp: 11/25/2003 13:09:23.101541
- 16) Table Event ...
Table schema: RIIHI
Table name: STAFF
- Record is the result of a flush: FALSE
Table type: User
Data object pages: 1
Rows read: 36
Data object pages: 1
Rows read: 36
Rows written: 1
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 2
Table event timestamp: 11/25/2003 13:09:23.101890
- 17) Table Event ...
Table schema: RIIHI
Table name: DEPARTMENT

Event monitors

Record is the result of a flush: FALSE
Table type: User
Data object pages: 1
Rows read: 9
Rows written: 1
Overflow Accesses: 0
Page reorgs: 0
Tablespace id: 2
Table event timestamp: 11/25/2003 13:09:23.101918

Related concepts:

- “Event monitors” on page 59
- “Event type mappings to logical data groups” on page 129

Related tasks:

- “Collecting information about database system events” on page 61

Related reference:

- “Event types” on page 60

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.

Related concepts:

- “Event monitor self-describing data stream” on page 91
- “Event monitors” on page 59

Related tasks:

- “Collecting information about database system events” on page 61

Related reference:

- “Event monitor sample output” on page 80

Event monitor self-describing data stream

The output of an event monitor is a binary stream of logical data groupings that are exactly the same for both pipe and file event monitors. You can format the data stream either by using the `db2evmon` command or by developing a client application. This data stream is presented in a self-describing format. Figure 3 shows the structure of the data stream and Table 12 on page 92 provides some examples of the logical data groups and monitor elements that could be returned.

Note: In the examples and tables descriptive names are used for the identifiers. These names are prefixed by `SQLM_ELM_` in the actual data stream. For example, `db_event` would appear as `SQLM_ELM_DB_EVENT` in the event monitor output. Types are prefixed with `SQLM_TYPE_` in the actual data stream. For example, headers appear as `SQLM_TYPE_HEADER` in the data stream.

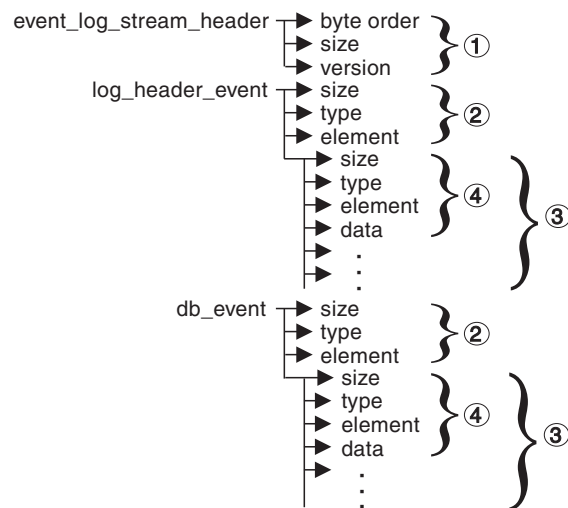


Figure 3. Event Monitor Data Stream

1. The structure of the `sqlm_event_log_data_stream_header` is different than the other headers in the data stream. The version field determines if the output can be processed as a Version 8 data stream.

This header has the same size and type as pre-Version 6 event monitor streams. This allows applications to determine if the event monitor output is self-describing or is in the pre-Version 6 static format.

Note: This monitor element is extracted by reading `sizeof(sqlm_event_log_data_stream)` bytes from the data stream.

2. Each logical data group begins with a header that indicates its size and element name. This does not apply `event_log_stream_header`, as its size element contains a dummy value to maintain backwards compatibility.
3. The size element in the header indicates the size of all the data in that logical data group.

Event monitors

4. Monitor element information follows its logical data group header and is also self-describing.

Table 12. Sample event data stream

Logical Data Group	Data Stream	Description
event_log_stream_header	sqlm_little_endian	Not used (for compatibility with previous releases).
	200	Not used (for compatibility with previous releases).
	sqlm_dbmon_version8	The version of the database manager that returned the data. Only Version 6, Version 7, and Version 8 monitors can write data in the self-describing format.
log_header_event	100	Size of the logical data group.
	header	Indicates the start of a logical data group.
	log_header	Name of the logical data group.
	4	Size of the data stored in this monitor element.
	u32bit	Monitor element type - 32 bit numeric.
	byte_order	The name of the monitor element collected.
	little_endian	The collected value for this element.
db_event	2	Size of the data stored in this monitor element.
	u16bit	Monitor element type - unsigned 16 bit numeric.
	codepage_id	The name of the monitor element collected.
	850	The collected value for this element.
db_event	100	Size of the logical data group.
	header	Indicates the start of a logical data group.
	db_event	Name of the logical data group.
	4	Size of the data stored in this monitor element.
	u32bit	Monitor element type - unsigned 32 bit numeric.
lock_waits	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. Only Version 6, Version 7, and Version 8 monitors write their data in the self-describing format. If you are working with a monitor from one of these versions, you can start processing the self-describing data stream. 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 codepage 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). Codepage 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.

Related concepts:

- “Event monitor file management” on page 73
- “Event monitor named pipe management” on page 76
- “Event type mappings to logical data groups” on page 129

Related tasks:

- “Transferring event monitor data between systems” on page 93

Related reference:

- “Event monitor sample output” on page 80

Related samples:

- “bldevm -- Builds the event monitor program, evm, on AIX (C)”
- “bldevm.bat -- Builds event monitor program, evm, on Windows”
- “evmprint.c -- Prints all events generated by an event monitor on AIX (C)”
- “evmprint.c -- Prints all events generated by an event monitor on Windows (C)”
- “evm.c -- Process event monitor data on AIX (C)”
- “evm.c -- Process event monitor data on Windows (C)”
- “evmread.c -- Read the event monitor self describing data stream on AIX (C)”
- “evmread.c -- Read the event monitor self describing data stream on Windows (C)”

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.

Procedure:

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;
```

Event monitors

```
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 */
```

Related concepts:

- “Event monitor file management” on page 73
- “Event monitor named pipe management” on page 76
- “Event monitor self-describing data stream” on page 91

Part 2. System Monitor Reference

Chapter 5. System Monitor 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.

Notes:

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 13. 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
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

System Monitor Logical data groups

Table 13. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

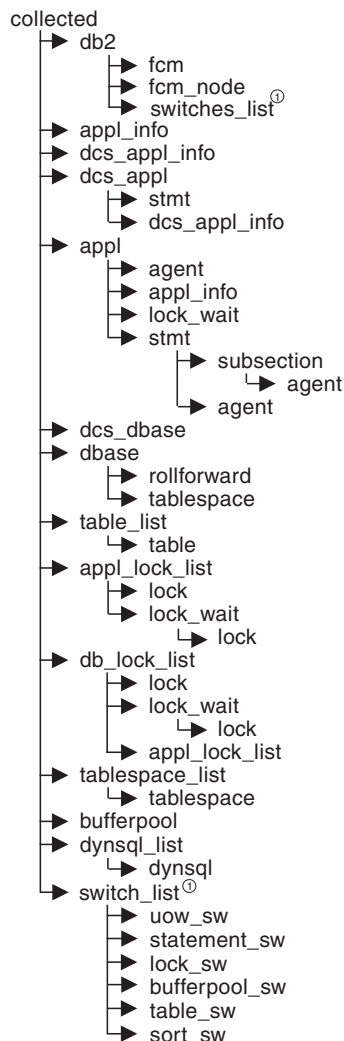
API request type	CLP command	SQL administrative view	Logical data groups
SQLMA_DBASE_ALL	get snapshot for all databases	SNAPDB	dbase
		SNAPSTORAGE_PATHS	db_storage_group
			rollforward
		SNAPTbsp	tablespace
		SNAPDB_MEMORY_POOL	memory_pool
	list active databases		dbase
SQLMA_DCS_DBASE	get snapshot for 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
		SNAPSUBSECTION	subsection
		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
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>		dcs_appl, dcs_stmt, dcs_appl_info, stmt_transmissions

Table 13. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

API request type	CLP command	SQL administrative view	Logical data groups
SQLMA_DCS_APPL_ALL	get snapshot for all dcs applications		dc_s_appl, dc_s_stmt, dc_s_appl_info, stmt_transmissions
SQLMA_DCS_APPL_HANDLE	get snapshot for dcs application agentid <i>appl-handle</i>		dc_s_appl, dc_s_stmt, dc_s_appl_info, stmt_transmissions
SQLMA_DCS_DBASE_APPLS	get snapshot for dcs applications on <i>dbname</i>		dc_s_appl, dc_s_stmt, dc_s_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
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.

System Monitor Logical data groups



^①Similar 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.

Related reference:

- "Supported administrative SQL routines and views" in *Administrative SQL Routines and Views*
- "GET SNAPSHOT command" in *Command Reference*

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 14. Snapshot Monitor Logical Data Groups and Monitor Elements

Snapshot logical data groups	Monitor element
agent	"agent_pid - Process or Thread ID " on page 189
	"lock_timeout_val - Lock timeout " on page 321

Table 14. 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 371
	"agent_sys_cpu_time - System CPU Time used by Agent " on page 413
	"agent_usr_cpu_time - User CPU Time used by Agent " on page 413
	"agents_stolen - Stolen Agents " on page 199
	"appl_con_time - Connection Request Start Timestamp " on page 183
	"appl_idle_time - Application Idle Time " on page 187
	"appl_priority - Application Agent Priority " on page 179
	"appl_priority_type - Application Priority Type " on page 180
	"associated_agents_top - Maximum Number of Associated Agents " on page 200
	"authority_lvl - User Authorization Level " on page 180
	"binds_precompiles - Binds/Precompiles Attempted " on page 383
	"cat_cache_inserts - Catalog Cache Inserts " on page 274
	"cat_cache_lookups - Catalog Cache Lookups " on page 273
	"cat_cache_overflows - Catalog Cache Overflows " on page 275
	"commit_sql_stmts - Commit Statements Attempted " on page 375
	"conn_complete_time - Connection Request Completion Timestamp " on page 184
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements " on page 378
	"deadlocks - Deadlocks Detected " on page 303
	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_read_time - Direct Read Time " on page 272
	"direct_reads - Direct Reads From Database " on page 269
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_write_time - Direct Write Time " on page 272
	"direct_writes - Direct Writes to Database " on page 269
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted " on page 373
	"failed_sql_stmts - Failed Statement Operations " on page 374
	"hash_join_overflows - Hash Join Overflows " on page 218
	"hash_join_small_overflows - Hash Join Small Overflows " on page 218
	"inbound_comm_address - Inbound Communication Address " on page 448
	"int_auto_rebinds - Internal Automatic Rebinds " on page 379
	"int_commits - Internal Commits " on page 380
	"int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock " on page 382
	"int_rollbacks - Internal Rollbacks " on page 381
	"int_rows_deleted - Internal Rows Deleted " on page 358
	"int_rows_inserted - Internal Rows Inserted " on page 359
	"int_rows_updated - Internal Rows Updated " on page 358
	"last_reset - Last Reset Timestamp " on page 419
	"lock_escalation - Lock Escalation " on page 311
	"lock_timeouts - Number of Lock Timeouts " on page 309
	"lock_timeout_val - Lock timeout " on page 321
	"lock_wait_time - Time Waited On Locks " on page 319
	"lock_waits - Lock Waits " on page 318
	"locks_held - Locks Held " on page 302
"locks_waiting - Current Agents Waiting On Locks " on page 320	

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
appl (continued)	<p>"num_agents - Number of Agents Working on a Statement " on page 411</p> <p>"open_loc_curs - Open Local Cursors " on page 371</p> <p>"open_loc_curs_blk - Open Local Cursors with Blocking " on page 372</p> <p>"open_rem_curs - Open Remote Cursors " on page 369</p> <p>"open_rem_curs_blk - Open Remote Cursors with Blocking " on page 369</p> <p>"pkg_cache_inserts - Package Cache Inserts " on page 279</p> <p>"pkg_cache_lookups - Package Cache Lookups " on page 277</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235</p> <p>"pool_data_writes - Buffer Pool Data Writes " on page 237</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240</p> <p>"pool_index_writes - Buffer Pool Index Writes " on page 241</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time " on page 243</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236</p> <p>"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239</p> <p>"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241</p> <p>"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads " on page 265</p> <p>"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads " on page 266</p> <p>"pool_write_time - Total Buffer Pool Physical Write Time " on page 243</p> <p>"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads " on page 262</p> <p>"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads " on page 263</p> <p>"pool_xda_writes - Buffer Pool XDA Data Writes " on page 264</p> <p>"prefetch_wait_time - Time Waited for Prefetch " on page 256</p> <p>"prev_uow_stop_time - Previous Unit of Work Completion Timestamp " on page 184</p> <p>"priv_workspace_num_overflows - Private Workspace Overflows " on page 285</p> <p>"priv_workspace_section_inserts - Private Workspace Section Inserts " on page 287</p> <p>"priv_workspace_section_lookups - Private Workspace Section Lookups " on page 286</p> <p>"priv_workspace_size_top - Maximum Private Workspace Size " on page 285</p> <p>"rej_curs_blk - Rejected Block Cursor Requests " on page 370</p> <p>"rollback_sql_stmts - Rollback Statements Attempted " on page 376</p> <p>"rows_deleted - Rows Deleted " on page 353</p> <p>"rows_inserted - Rows Inserted " on page 353</p> <p>"rows_read - Rows Read " on page 356</p> <p>"rows_selected - Rows Selected " on page 355</p> <p>"rows_updated - Rows Updated " on page 354</p> <p>"rows_written - Rows Written " on page 355</p> <p>"select_sql_stmts - Select SQL Statements Executed " on page 377</p> <p>"shr_workspace_num_overflows - Shared Workspace Overflows " on page 283</p> <p>"shr_workspace_section_inserts - Shared Workspace Section Inserts " on page 284</p> <p>"shr_workspace_section_lookups - Shared Workspace Section Lookups " on page 283</p> <p>"shr_workspace_size_top - Maximum Shared Workspace Size " on page 282</p> <p>"sort_overflows - Sort Overflows " on page 211</p> <p>"sql_reqs_since_commit - SQL Requests Since Last Commit " on page 382</p> <p>"static_sql_stmts - Static SQL Statements Attempted " on page 373</p> <p>"xquery_stmts - XQuery Statements Attempted" on page 383</p>

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
appl (continued)	"total_hash_joins - Total Hash Joins " on page 215
	"total_hash_loops - Total Hash Loops " on page 217
	"total_sort_time - Total Sort Time " on page 210
	"total_sorts - Total Sorts " on page 210
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed " on page 378
	"unread_prefetch_pages - Unread Prefetch Pages " on page 256
	"uow_comp_status - Unit of Work Completion Status " on page 186
	"uow_elapsed_time - Most Recent Unit of Work Elapsed Time " on page 186
	"uow_lock_wait_time - Total Time Unit of Work Waited on Locks " on page 320
	"uow_log_space_used - Unit of Work Log Space Used " on page 292
	"uow_start_time - Unit of Work Start Timestamp " on page 185
	"uow_stop_time - Unit of Work Stop Timestamp " on page 185
	"x_lock_escals - Exclusive Lock Escalations " on page 305
	appl_id_info
"appl_id - Application ID " on page 169	
"appl_name - Application Name " on page 168	
"appl_status - Application Status " on page 164	
"auth_id - Authorization ID " on page 172	
"client_db_alias - Database Alias Used by Application " on page 174	
"client_nname - Configuration NNAME of Client " on page 173	
"client_prdid - Client Product/Version ID " on page 174	
"codepage_id - ID of Code Page Used by Application " on page 166	
"db_name - Database Name " on page 154	
"db_path - Database Path " on page 155	
"input_db_alias - Input Database Alias " on page 420	
"sequence_no - Sequence Number " on page 172	
"session_auth_id - Session Authorization ID " on page 173	
"status_change_time - Application Status Change Time " on page 167	

System Monitor Logical data groups

Table 14. 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 163	
	"appl_id - Application ID " on page 169	
	"appl_name - Application Name " on page 168	
	"appl_section_inserts - Section Inserts monitor element" on page 281	
	"appl_section_lookups - Section Lookups " on page 280	
	"appl_status - Application Status " on page 164	
	"auth_id - Authorization ID " on page 172	
	"authority_lvl - User Authorization Level " on page 180	
	"client_db_alias - Database Alias Used by Application " on page 174	
	"client_nname - Configuration NNAME of Client " on page 173	
	"client_pid - Client Process ID " on page 177	
	"client_platform - Client Operating Platform " on page 178	
	"client_prdid - Client Product/Version ID " on page 174	
	"client_protocol - Client Communication Protocol " on page 178	
	"codepage_id - ID of Code Page Used by Application " on page 166	
	"coord_agent_pid - Coordinator Agent " on page 189	
	"coord_node - Coordinating Node " on page 182	
	"corr_token - DRDA Correlation Token " on page 177	
	"db_name - Database Name " on page 154	
	"db_path - Database Path " on page 155	
	"execution_id - User Login ID " on page 176	
	"input_db_alias - Input Database Alias " on page 420	
	"num_assoc_agents - Number of Associated Agents " on page 201	
	"sequence_no - Sequence Number " on page 172	
	"status_change_time - Application Status Change Time " on page 167	
	"territory_code - Database Territory Code " on page 179	
	"tpmon_acc_str - TP Monitor Client Accounting String " on page 474	
	"tpmon_client_app - TP Monitor Client Application Name " on page 473	
	"tpmon_client_userid - TP Monitor Client User ID " on page 472	
	"tpmon_client_wkstn - TP Monitor Client Workstation Name " on page 473	
	appl_lock_list	"agent_id - Application Handle (agent ID) " on page 163
		"appl_id - Application ID " on page 169
"appl_name - Application Name " on page 168		
"appl_status - Application Status " on page 164		
"auth_id - Authorization ID " on page 172		
"client_db_alias - Database Alias Used by Application " on page 174		
"codepage_id - ID of Code Page Used by Application " on page 166		
"locks_held - Locks Held " on page 302		
"locks_waiting - Current Agents Waiting On Locks " on page 320		
"lock_wait_time - Time Waited On Locks " on page 319		
"sequence_no - Sequence Number " on page 172		
"session_auth_id - Session Authorization ID " on page 173		
"status_change_time - Application Status Change Time " on page 167		

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
appl_remote	<p>"commit_sql_stmts - Commit Statements Attempted " on page 375</p> <p>"create_nickname - Create Nicknames " on page 477</p> <p>"create_nickname_time - Create Nickname Response Time " on page 481</p> <p>"datasource_name - Data Source Name " on page 475</p> <p>"db_name - Database Name " on page 154</p> <p>"delete_sql_stmts - Deletes " on page 476</p> <p>"delete_time - Delete Response Time " on page 481</p> <p>"failed_sql_stmts - Failed Statement Operations " on page 374</p> <p>"insert_sql_stmts - Inserts " on page 475</p> <p>"insert_time - Insert Response Time " on page 480</p> <p>"passthru_time - Pass-Through Time " on page 482</p> <p>"passthru - Pass-Through " on page 477</p> <p>"remote_lock_time - Remote Lock Time " on page 483</p> <p>"remote_locks - Remote Locks " on page 478</p> <p>"rollback_sql_stmts - Rollback Statements Attempted " on page 376</p> <p>"rows_deleted - Rows Deleted " on page 353</p> <p>"rows_inserted - Rows Inserted " on page 353</p> <p>"rows_selected - Rows Selected " on page 355</p> <p>"rows_updated - Rows Updated " on page 354</p> <p>"select_sql_stmts - Select SQL Statements Executed " on page 377</p> <p>"select_time - Query Response Time " on page 479</p> <p>"sp_rows_selected - Rows Returned by Stored Procedures " on page 479</p> <p>"stored_proc_time - Stored Procedure Time " on page 482</p> <p>"stored_procs - Stored Procedures " on page 478</p> <p>"update_sql_stmts - Updates " on page 476</p> <p>"update_time - Update Response Time " on page 480</p>

System Monitor Logical data groups

Table 14. 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 258
	"bp_name - Buffer Pool Name " on page 255
	"bp_id - Buffer Pool ID " on page 232
	"db_name - Database Name " on page 154
	"db_path - Database Path " on page 155
	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_reads - Direct Reads From Database " on page 269
	"direct_read_time - Direct Read Time " on page 272
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_writes - Direct Writes to Database " on page 269
	"direct_write_time - Direct Write Time " on page 272
	"files_closed - Database Files Closed " on page 244
	"input_db_alias - Input Database Alias " on page 420
	"pages_from_block_ios - Total Number of Pages Read by Block IO " on page 259
	"pages_from_vectored_ios - Total Number of Pages Read by Vectored IO " on page 257
	"physical_page_maps - Number of Physical Page Maps " on page 259
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests " on page 250
	"pool_async_data_reads - Buffer Pool Asynchronous Data Reads " on page 245
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes " on page 246
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests " on page 251
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads " on page 248
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes " on page 247
	"pool_async_read_time - Buffer Pool Asynchronous Read Time " on page 249
	"pool_async_write_time - Buffer Pool Asynchronous Write Time " on page 249
	"pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests " on page 259
	"pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads " on page 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 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_data_writes - Buffer Pool Data Writes " on page 237
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_index_writes - Buffer Pool Index Writes " on page 241
	"pool_read_time - Total Buffer Pool Physical Read Time " on page 243
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers " on page 254
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads " on page 265
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads " on page 266
	"pool_write_time - Total Buffer Pool Physical Write Time " on page 243
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads " on page 262
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads " on page 263
	"pool_xda_writes - Buffer Pool XDA Data Writes " on page 264
	"vectored_ios - Number of Vectored IO Requests " on page 257

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
bufferpool_nodeinfo	"bp_cur_buffsz - Current Size of Buffer Pool " on page 267 "bp_new_buffsz - New Buffer Pool Size " on page 268 "bp_pages_left_to_remove - Number of Pages Left to Remove " on page 268 "bp_tbsp_use_count - Number of Table Spaces Mapped to Buffer Pool " on page 268 "node_number - Node Number " on page 181
collected	"node_number - Node Number " on page 181 "server_db2_type - Database Manager Type at Monitored (Server) Node " on page 150 "server_instance_name - Server Instance Name " on page 149 "server_nname - Configuration NNAME at Monitoring (Server) Database Partition " on page 149 "server_prdid - Server Product/Version ID " on page 150 "server_version - Server Version " on page 151 switch_list Monitor switches control data "time_stamp - Snapshot Time " on page 420 "time_zone_disp - Time Zone Displacement " on page 153

System Monitor Logical data groups

Table 14. 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 198	
	"agents_from_pool - Agents Assigned From Pool " on page 198	
	"agents_registered - Agents Registered " on page 196	
	"agents_registered_top - Maximum Number of Agents Registered " on page 196	
	"agents_stolen - Stolen Agents " on page 199	
	"agents_waiting_on_token - Agents Waiting for a Token " on page 196	
	"agents_waiting_top - Maximum Number of Agents Waiting " on page 197	
	"comm_private_mem - Committed Private Memory " on page 200	
	"con_local_databases - Local Databases with Current Connects " on page 193	
	"coord_agents_top - Maximum Number of Coordinating Agents " on page 199	
	"db2start_time - Start Database Manager Timestamp " on page 148	
	"db_status - Status of Database " on page 156	
	"gw_total_cons - Total Number of Attempted Connections for DB2 Connect " on page 441	
	"gw_cur_cons - Current Number of Connections for DB2 Connect " on page 442	
	"gw_cons_wait_host - Number of Connections Waiting for the Host to Reply " on page 442	
	"gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request " on page 443	
	"idle_agents - Number of Idle Agents " on page 197	
	"last_reset - Last Reset Timestamp " on page 419	
	"local_cons - Local Connections " on page 192	
	"local_cons_in_exec - Local Connections Executing in the Database Manager " on page 193	
	"max_agent_overflows - Maximum Agent Overflows " on page 201	
	"num_gw_conn_switches - Connection Switches " on page 202	
	"num_nodes_in_db2_instance - Number of Nodes in Partition " on page 420	
	"piped_sorts_requested - Piped Sorts Requested " on page 208	
	"piped_sorts_accepted - Piped Sorts Accepted " on page 209	
	"post_threshold_hash_joins - Hash Join Threshold " on page 216	
	"post_threshold_sorts - Post Threshold Sorts " on page 208	
	"product_name - Product Name " on page 152	
	"rem_cons_in - Remote Connections To Database Manager " on page 191	
	"rem_cons_in_exec - Remote Connections Executing in the Database Manager " on page 192	
	"service_level - Service Level " on page 151	
	"smallest_log_avail_node - Node with Least Available Log Space " on page 168	
	"sort_heap_allocated - Total Sort Heap Allocated " on page 207	
	"sort_heap_top - Sort Private Heap High Water Mark " on page 213	
	switch_list Monitor switches control data	
	db_lock_list	"apps_cur_cons - Applications Connected Currently " on page 194
		"db_name - Database Name " on page 154
		"db_path - Database Path " on page 155
		"input_db_alias - Input Database Alias " on page 420
		"locks_held - Locks Held " on page 302
"locks_waiting - Current Agents Waiting On Locks " on page 320		

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dbase	"active_hash_joins - Active hash joins " on page 215
	"active_sorts - Active Sorts " on page 212
	"agents_top - Number of Agents Created " on page 412
	"appl_id_oldest_xact - Application with Oldest Transaction " on page 167
	"appl_section_inserts - Section Inserts monitor element" on page 281
	"appl_section_lookups - Section Lookups " on page 280
	"appls_cur_cons - Applications Connected Currently " on page 194
	"appls_in_db2 - Applications Executing in the Database Currently " on page 195
	"binds_precompiles - Binds/Precompiles Attempted " on page 383
	"cat_cache_inserts - Catalog Cache Inserts " on page 274
	"cat_cache_lookups - Catalog Cache Lookups " on page 273
	"cat_cache_overflows - Catalog Cache Overflows " on page 275
	"cat_cache_size_top - Catalog Cache High Water Mark " on page 276
	"catalog_node - Catalog Node Number " on page 158
	"catalog_node_name - Catalog Node Network Name " on page 157
	"commit_sql_stmts - Commit Statements Attempted " on page 375
	"connections_top - Maximum Number of Concurrent Connections " on page 183
	"coord_agents_top - Maximum Number of Coordinating Agents " on page 199
	"db_conn_time - Database Activation Timestamp " on page 155
	"db_heap_top - Maximum Database Heap Allocated " on page 288
	"db_location - Database Location " on page 157
	"db_name - Database Name " on page 154
	"db_path - Database Path " on page 155
	"db_status - Status of Database " on page 156
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements " on page 378
	"deadlocks - Deadlocks Detected " on page 303
	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_read_time - Direct Read Time " on page 272
	"direct_reads - Direct Reads From Database " on page 269
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_write_time - Direct Write Time " on page 272
	"direct_writes - Direct Writes to Database " on page 269
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted " on page 373
	"failed_sql_stmts - Failed Statement Operations " on page 374
	"files_closed - Database Files Closed " on page 244

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dbase (continued)	<p>"hash_join_overflows - Hash Join Overflows " on page 218</p> <p>"hash_join_small_overflows - Hash Join Small Overflows " on page 218</p> <p>"input_db_alias - Input Database Alias " on page 420</p> <p>"int_auto_rebinds - Internal Automatic Rebinds " on page 379</p> <p>"int_commits - Internal Commits " on page 380</p> <p>"int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock " on page 382</p> <p>"int_rollbacks - Internal Rollbacks " on page 381</p> <p>"int_rows_deleted - Internal Rows Deleted " on page 358</p> <p>"int_rows_inserted - Internal Rows Inserted " on page 359</p> <p>"int_rows_updated - Internal Rows Updated " on page 358</p> <p>"last_backup - Last Backup Timestamp " on page 158</p> <p>"last_reset - Last Reset Timestamp " on page 419</p> <p>"lock_escals - Number of Lock Escalations " on page 304</p> <p>"lock_list_in_use - Total Lock List Memory In Use " on page 302</p> <p>"lock_timeouts - Number of Lock Timeouts " on page 309</p> <p>"lock_wait_time - Time Waited On Locks " on page 319</p> <p>"lock_waits - Lock Waits " on page 318</p> <p>"locks_held - Locks Held " on page 302</p> <p>"locks_waiting - Current Agents Waiting On Locks " on page 320</p> <p>"log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages " on page 294</p> <p>"log_read_time - Log Read Time " on page 296</p> <p>"log_reads - Number of Log Pages Read " on page 291</p> <p>"log_to_redo_for_recovery - Amount of Log to be Redone for Recovery " on page 295</p> <p>"log_write_time - Log Write Time " on page 295</p> <p>"log_writes - Number of Log Pages Written " on page 291</p> <p>"num_assoc_agents - Number of Associated Agents " on page 201</p> <p>"num_db_storage_paths - Number of automatic storage paths " on page 159</p> <p>"num_indoubt_trans - Number of Indoubt Transactions " on page 317</p> <p>"num_log_buffer_full - Number of Full Log Buffers " on page 298</p> <p>"num_log_data_found_in_buffer - Number of Log Data Found In Buffer " on page 298</p> <p>"num_log_part_page_io - Number of Partial Log Page Writes " on page 297</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 296</p> <p>"pkg_cache_inserts - Package Cache Inserts " on page 279</p> <p>"pkg_cache_lookups - Package Cache Lookups " on page 277</p> <p>"pkg_cache_num_overflows - Package Cache Overflows " on page 279</p> <p>"pkg_cache_size_top - Package Cache High Water Mark " on page 280</p> <p>"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests " on page 250</p> <p>"pool_async_data_reads - Buffer Pool Asynchronous Data Reads " on page 245</p> <p>"pool_async_data_writes - Buffer Pool Asynchronous Data Writes " on page 246</p> <p>"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests " on page 251</p> <p>"pool_async_index_reads - Buffer Pool Asynchronous Index Reads " on page 248</p> <p>"pool_async_index_writes - Buffer Pool Asynchronous Index Writes " on page 247</p> <p>"pool_async_read_time - Buffer Pool Asynchronous Read Time " on page 249</p> <p>"pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests " on page 259</p> <p>"pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads " on page 260</p> <p>"pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes " on page 261</p>

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dbase (continued)	<p>"pool_async_write_time - Buffer Pool Asynchronous Write Time " on page 249</p> <p>"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233</p> <p>"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235</p> <p>"pool_data_writes - Buffer Pool Data Writes " on page 237</p> <p>"pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered " on page 252</p> <p>"pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered " on page 254</p> <p>"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238</p> <p>"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240</p> <p>"pool_index_writes - Buffer Pool Index Writes " on page 241</p> <p>"pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered " on page 251</p> <p>"pool_no_victim_buffer - Buffer Pool No Victim Buffers " on page 254</p> <p>"pool_read_time - Total Buffer Pool Physical Read Time " on page 243</p> <p>"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234</p> <p>"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236</p> <p>"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239</p> <p>"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241</p> <p>"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads " on page 265</p> <p>"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads " on page 266</p> <p>"pool_write_time - Total Buffer Pool Physical Write Time " on page 243</p> <p>"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads " on page 262</p> <p>"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads " on page 263</p> <p>"pool_xda_writes - Buffer Pool XDA Data Writes " on page 264</p> <p>"post_shrthreshold_hash_joins - Post threshold hash joins " on page 216</p> <p>"post_shrthreshold_sorts - Post threshold sorts " on page 214</p> <p>"priv_workspace_num_overflows - Private Workspace Overflows " on page 285</p> <p>"priv_workspace_section_inserts - Private Workspace Section Inserts " on page 287</p> <p>"priv_workspace_section_lookups - Private Workspace Section Lookups " on page 286</p> <p>"priv_workspace_size_top - Maximum Private Workspace Size " on page 285</p> <p>"rollback_sql_stmts - Rollback Statements Attempted " on page 376</p> <p>"rows_deleted - Rows Deleted " on page 353</p> <p>"rows_inserted - Rows Inserted " on page 353</p> <p>"rows_read - Rows Read " on page 356</p> <p>"rows_selected - Rows Selected " on page 355</p> <p>"rows_updated - Rows Updated " on page 354</p> <p>"sec_log_used_top - Maximum Secondary Log Space Used " on page 289</p> <p>"sec_logs_allocated - Secondary Logs Allocated Currently " on page 290</p> <p>"select_sql_stmts - Select SQL Statements Executed " on page 377</p> <p>"server_platform - Server Operating System " on page 152</p> <p>"shr_workspace_num_overflows - Shared Workspace Overflows " on page 283</p> <p>"shr_workspace_section_inserts - Shared Workspace Section Inserts " on page 284</p> <p>"shr_workspace_section_lookups - Shared Workspace Section Lookups " on page 283</p> <p>"shr_workspace_size_top - Maximum Shared Workspace Size " on page 282</p> <p>"sort_heap_allocated - Total Sort Heap Allocated " on page 207</p> <p>"sort_overflows - Sort Overflows " on page 211</p> <p>"sort_shrheap_allocated - Sort Share Heap Currently Allocated " on page 213</p> <p>"sort_shrheap_top - Sort Share Heap High Water Mark " on page 213</p>

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dbase (continued)	"static_sql_stmts - Static SQL Statements Attempted " on page 373
	"tot_log_used_top - Maximum Total Log Space Used " on page 290
	"total_cons - Connects Since Database Activation " on page 194
	"total_hash_joins - Total Hash Joins " on page 215
	"total_hash_loops - Total Hash Loops " on page 217
	"total_log_available - Total Log Available " on page 293
	"total_log_used - Total Log Space Used " on page 292
	"total_sec_cons - Secondary Connections " on page 200
	"total_sort_time - Total Sort Time " on page 210
	"total_sorts - Total Sorts " on page 210
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed " on page 378
	"unread_prefetch_pages - Unread Prefetch Pages " on page 256
	"x_lock_escals - Exclusive Lock Escalations " on page 305
	"xquery_stmts - XQuery Statements Attempted" on page 383
dbase_remote	"commit_sql_stmts - Commit Statements Attempted " on page 375
	"create_nickname - Create Nicknames " on page 477
	"create_nickname_time - Create Nickname Response Time " on page 481
	"datasource_name - Data Source Name " on page 475
	"db_name - Database Name " on page 154
	"delete_sql_stmts - Deletes " on page 476
	"delete_time - Delete Response Time " on page 481
	"disconnects - Disconnects " on page 475
	"failed_sql_stmts - Failed Statement Operations " on page 374
	"insert_sql_stmts - Inserts " on page 475
	"insert_time - Insert Response Time " on page 480
	"passthru_time - Pass-Through Time " on page 482
	"passthru - Pass-Through " on page 477
	"remote_lock_time - Remote Lock Time " on page 483
	"remote_locks - Remote Locks " on page 478
	"rollback_sql_stmts - Rollback Statements Attempted " on page 376
	"rows_deleted - Rows Deleted " on page 353
	"rows_inserted - Rows Inserted " on page 353
	"rows_selected - Rows Selected " on page 355
	"rows_updated - Rows Updated " on page 354
	"select_sql_stmts - Select SQL Statements Executed " on page 377
	"select_time - Query Response Time " on page 479
	"sp_rows_selected - Rows Returned by Stored Procedures " on page 479
	"stored_proc_time - Stored Procedure Time " on page 482
	"stored_procs - Stored Procedures " on page 478
	"total_cons - Connects Since Database Activation " on page 194
	"update_sql_stmts - Updates " on page 476
	"update_time - Update Response Time " on page 480

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
db_storage_group	<p>"db_storage_path - Automatic storage path " on page 159</p> <p>"sto_path_free_sz - Automatic Storage Path Free Space " on page 159</p> <p>"fs_used_size - Amount of Space Used on a File System " on page 160</p> <p>"fs_total_size - Total Size of a File System " on page 160</p> <p>"fs_id - Unique File System Identification Number " on page 161</p> <p>"fs_type - File System Type " on page 162</p> <p>"node_number - Node Number " on page 181</p>
dcs_appl	<p>"appl_idle_time - Application Idle Time " on page 187</p> <p>"commit_sql_stmts - Commit Statements Attempted " on page 375</p> <p>"elapsed_exec_time - Statement Execution Elapsed Time " on page 467</p> <p>"failed_sql_stmts - Failed Statement Operations " on page 374</p> <p>"gw_con_time - DB2 Connect Gateway First Connect Initiated " on page 441</p> <p>"gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing " on page 443</p> <p>"host_response_time - Host Response Time " on page 468</p> <p>"inbound_bytes_received - Inbound Number of Bytes Received " on page 449</p> <p>"inbound_bytes_sent - Inbound Number of Bytes Sent " on page 450</p> <p>"last_reset - Last Reset Timestamp " on page 419</p> <p>"max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes " on page 453</p> <p>"max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes " on page 454</p> <p>"max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes " on page 455</p> <p>"max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes " on page 456</p> <p>"max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes " on page 457</p> <p>"max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes " on page 458</p> <p>"max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes " on page 459</p> <p>"max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes " on page 460</p> <p>"max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes " on page 461</p> <p>"max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes " on page 462</p> <p>"max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes " on page 462</p>

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dcs_appl (continued)	<p>"max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes " on page 453</p> <p>"max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes " on page 453</p> <p>"max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes " on page 454</p> <p>"max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes " on page 455</p> <p>"max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes " on page 456</p> <p>"max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes " on page 457</p> <p>"max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes " on page 458</p> <p>"max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes " on page 459</p> <p>"max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes " on page 460</p> <p>"max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes " on page 461</p> <p>"max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes " on page 462</p> <p>"max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms " on page 463</p> <p>"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms " on page 463</p> <p>"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms " on page 464</p> <p>"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms " on page 465</p> <p>"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms " on page 465</p> <p>"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms " on page 466</p> <p>"network_time_top - Maximum Network Time for Statement " on page 466</p> <p>"network_time_bottom - Minimum Network Time for Statement " on page 467</p> <p>"open_cursors - Number of Open Cursors " on page 445</p> <p>"outbound_bytes_received - Outbound Number of Bytes Received " on page 450</p> <p>"outbound_bytes_sent - Outbound Number of Bytes Sent " on page 449</p> <p>"prev_uow_stop_time - Previous Unit of Work Completion Timestamp " on page 184</p> <p>"rollback_sql_stmts - Rollback Statements Attempted " on page 376</p> <p>"rows_selected - Rows Selected " on page 355</p> <p>"sql_stmts - Number of SQL Statements Attempted " on page 444</p> <p>"tpmon_acc_str - TP Monitor Client Accounting String " on page 474</p> <p>"tpmon_client_app - TP Monitor Client Application Name " on page 473</p> <p>"tpmon_client_userid - TP Monitor Client User ID " on page 472</p> <p>"tpmon_client_wkstn - TP Monitor Client Workstation Name " on page 473</p> <p>"uow_comp_status - Unit of Work Completion Status " on page 186</p> <p>"uow_elapsed_time - Most Recent Unit of Work Elapsed Time " on page 186</p> <p>"uow_start_time - Unit of Work Start Timestamp " on page 185</p> <p>"uow_stop_time - Unit of Work Stop Timestamp " on page 185</p> <p>"xid - Transaction ID " on page 467</p>

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dcs_appl_info	"agent_id - Application Handle (agent ID) " on page 163
	"agent_status - DCS Application Agents " on page 446
	"appl_id - Application ID " on page 169
	"appl_name - Application Name " on page 168
	"auth_id - Authorization ID " on page 172
	"client_nname - Configuration NNAME of Client " on page 173
	"client_pid - Client Process ID " on page 177
	"client_platform - Client Operating Platform " on page 178
	"client_prdid - Client Product/Version ID " on page 174
	"client_protocol - Client Communication Protocol " on page 178
	"codepage_id - ID of Code Page Used by Application " on page 166
	"dcs_appl_status - DCS Application Status " on page 446
	"dcs_db_name - DCS Database Name " on page 440
	"execution_id - User Login ID " on page 176
	"gw_db_alias - Database Alias at the Gateway " on page 440
	"host_ccsid - Host Coded Character Set ID " on page 447
	"host_db_name - Host Database Name " on page 440
	"host_prdid - Host Product/Version ID " on page 175
	"inbound_comm_address - Inbound Communication Address " on page 448
	"outbound_appl_id - Outbound Application ID " on page 175
	"outbound_comm_address - Outbound Communication Address " on page 448
	"outbound_comm_protocol - Outbound Communication Protocol " on page 447
	"outbound_sequence_no - Outbound Sequence Number " on page 176
	"sequence_no - Sequence Number " on page 172
"status_change_time - Application Status Change Time " on page 167	

System Monitor Logical data groups

Table 14. 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 375</p> <p>"con_elapsed_time - Most Recent Connection Elapsed Time " on page 470</p> <p>"con_response_time - Most Recent Response Time for Connect " on page 470</p> <p>"dcs_db_name - DCS Database Name " on page 440</p> <p>"elapsed_exec_time - Statement Execution Elapsed Time " on page 467</p> <p>"failed_sql_stmts - Failed Statement Operations " on page 374</p> <p>"gw_comm_error_time - Communication Error Time " on page 471</p> <p>"gw_comm_errors - Communication Errors " on page 471</p> <p>"gw_con_time - DB2 Connect Gateway First Connect Initiated " on page 441</p> <p>"gw_connections_top - Maximum Number of Concurrent Connections to Host Database " on page 441</p> <p>"gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request " on page 443</p> <p>"gw_cons_wait_host - Number of Connections Waiting for the Host to Reply " on page 442</p> <p>"gw_cur_cons - Current Number of Connections for DB2 Connect " on page 442</p> <p>"gw_total_cons - Total Number of Attempted Connections for DB2 Connect " on page 441</p> <p>"host_db_name - Host Database Name " on page 440</p> <p>"host_response_time - Host Response Time " on page 468</p> <p>"inbound_bytes_received - Inbound Number of Bytes Received " on page 449</p> <p>"last_reset - Last Reset Timestamp " on page 419</p> <p>"max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes " on page 453</p> <p>"max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes " on page 454</p> <p>"max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes " on page 455</p> <p>"max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes " on page 456</p> <p>"max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes " on page 457</p> <p>"max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes " on page 458</p> <p>"max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes " on page 459</p> <p>"max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes " on page 460</p> <p>"max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes " on page 461</p> <p>"max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes " on page 462</p> <p>"max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes " on page 462</p>

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dcs_dbase (continued)	<p>"max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes " on page 453</p> <p>"max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes " on page 453</p> <p>"max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes " on page 454</p> <p>"max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes " on page 455</p> <p>"max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes " on page 456</p> <p>"max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes " on page 457</p> <p>"max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes " on page 458</p> <p>"max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes " on page 459</p> <p>"max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes " on page 460</p> <p>"max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes " on page 461</p> <p>"max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes " on page 462</p> <p>"max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms " on page 463</p> <p>"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms " on page 463</p> <p>"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms " on page 464</p> <p>"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms " on page 465</p> <p>"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms " on page 465</p> <p>"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms " on page 466</p> <p>"network_time_bottom - Minimum Network Time for Statement " on page 467</p> <p>"network_time_top - Maximum Network Time for Statement " on page 466</p> <p>"outbound_bytes_sent - Outbound Number of Bytes Sent " on page 449</p> <p>"rollback_sql_stmts - Rollback Statements Attempted " on page 376</p> <p>"rows_selected - Rows Selected " on page 355</p> <p>"sql_stmts - Number of SQL Statements Attempted " on page 444</p>

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dcs_stmt	"blocking_cursor - Blocking Cursor " on page 471
	"creator - Application Creator " on page 390
	"elapsed_exec_time - Statement Execution Elapsed Time " on page 467
	"fetch_count - Number of Successful Fetches " on page 394
	"gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing " on page 443
	"host_response_time - Host Response Time " on page 468
	"inbound_bytes_received - Inbound Number of Bytes Received " on page 449
	"inbound_bytes_sent - Inbound Number of Bytes Sent " on page 450
	"num_transmissions_group - Number of Transmissions Group " on page 469
	"num_transmissions - Number of Transmissions " on page 469
	"outbound_bytes_received - Outbound Number of Bytes Received " on page 450
	"outbound_bytes_sent - Outbound Number of Bytes Sent " on page 449
	"package_name - Package Name " on page 387
	"query_card_estimate - Query Number of Rows Estimate " on page 395
	"query_cost_estimate - Query Cost Estimate " on page 396
	"section_number - Section Number " on page 389
	"stmt_elapsed_time - Most Recent Statement Elapsed Time " on page 392
	"stmt_operation/operation - Statement Operation " on page 386
	"stmt_start - Statement Operation Start Timestamp " on page 391
	"stmt_stop - Statement Operation Stop Timestamp " on page 391
"stmt_text - SQL Dynamic Statement Text " on page 393	
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 181

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
dynsql	"int_rows_deleted - Internal Rows Deleted " on page 358
	"int_rows_inserted - Internal Rows Inserted " on page 359
	"int_rows_updated - Internal Rows Updated " on page 358
	"num_compilations - Statement Compilations " on page 410
	"num_executions - Statement Executions " on page 409
	"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads " on page 265
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads " on page 266
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads " on page 262
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads " on page 263
	"prep_time_best - Statement Best Preparation Time " on page 410
	"prep_time_worst - Statement Worst Preparation Time " on page 410
	"rows_read - Rows Read " on page 356
	"rows_written - Rows Written " on page 355
	"sort_overflows - Sort Overflows " on page 211
	"stmt_sorts - Statement Sorts " on page 394
	"stmt_text - SQL Dynamic Statement Text " on page 393
	"total_exec_time - Elapsed Statement Execution Time " on page 411
	"total_sort_time - Total Sort Time " on page 210
"total_sys_cpu_time - Total System CPU for a Statement " on page 418	
"total_usr_cpu_time - Total User CPU for a Statement " on page 419	
dynsql_list	"db_name - Database Name " on page 154
	"db_path - Database Path " on page 155
fcm	"buff_free - FCM Buffers Currently Free " on page 219
	"buff_free_bottom - Minimum FCM Buffers Free " on page 219
	"ch_free - Channels Currently Free " on page 220
	"ch_free_bottom - Minimum Channels Free " on page 220
fcm_node	"connection_status - Connection Status " on page 221
	"node_number - Node Number " on page 181
	"total_buffers_sent - Total FCM Buffers Sent " on page 221
	"total_buffers_rcvd - Total FCM Buffers Received " on page 222

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
hadr	"hadr_connect_status - HADR Connection Status monitor element" on page 429
	"hadr_connect_time - HADR Connection Time monitor element" on page 430
	"hadr_heartbeat - HADR Heartbeat monitor element" on page 430
	"hadr_local_host - HADR Local Host monitor element" on page 431
	"hadr_local_service - HADR Local Service monitor element" on page 432
	"hadr_log_gap - HADR Log Gap " on page 437
	"hadr_primary_log_file - HADR Primary Log File monitor element" on page 434
	"hadr_primary_log_lsn - HADR Primary Log LSN monitor element" on page 435
	"hadr_primary_log_page - HADR Primary Log Page monitor element" on page 434
	"hadr_remote_host - HADR Remote Host monitor element" on page 432
	"hadr_remote_instance - HADR Remote Instance monitor element" on page 433
	"hadr_remote_service - HADR Remote Service monitor element" on page 433
	"hadr_role - HADR Role " on page 427
	"hadr_standby_log_file - HADR Standby Log File monitor element" on page 435
	"hadr_standby_log_lsn - HADR Standby Log LSN monitor element" on page 436
	"hadr_standby_log_page - HADR Standby Log Page monitor element" on page 436
	"hadr_state - HADR State monitor element" on page 428
"hadr_syncmode - HADR Synchronization Mode monitor element" on page 428	
"hadr_timeout - HADR Timeout monitor element" on page 433	
lock	"data_partition_id - Data Partition Identifier monitor element" on page 188
	"lock_attributes - Lock Attributes " on page 314
	"lock_count - Lock Count " on page 316
	"lock_current_mode - Original Lock Mode Before Conversion " on page 316
	"lock_escalation - Lock Escalation " on page 311
	"lock_hold_count - Lock Hold Count " on page 316
	"lock_mode - Lock Mode " on page 306
	"lock_name - Lock Name " on page 314
	"lock_object_name - Lock Object Name " on page 308
	"lock_object_type - Lock Object Type Waited On " on page 308
	"lock_release_flags - Lock Release Flags " on page 315
	"lock_status - Lock Status " on page 307
	"node_number - Node Number " on page 181
	"table_file_id - Table File ID " on page 360
"table_name - Table Name " on page 351	
"table_schema - Table Schema Name " on page 352	
"tablespace_name - Table Space Name " on page 328	

Table 14. 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 321</p> <p>"appl_id_holding_lk - Application ID Holding Lock " on page 322</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 311</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 314</p> <p>"lock_object_type - Lock Object Type Waited On " on page 308</p> <p>"lock_release_flags - Lock Release Flags " on page 315</p> <p>"lock_wait_start_time - Lock Wait Start Timestamp " on page 320</p> <p>"node_number - Node Number " on page 181</p> <p>"ss_number - Subsection Number " on page 404</p> <p>"table_name - Table Name " on page 351</p> <p>"table_schema - Table Schema Name " on page 352</p> <p>"tablespace_name - Table Space Name " on page 328</p> <p>"data_partition_id - Data Partition Identifier monitor element" on page 188</p>
memory_pool	<p>"node_number - Node Number " on page 181</p> <p>"pool_cur_size - Current Size of Memory Pool " on page 204</p> <p>"pool_id - Memory Pool Identifier " on page 203</p> <p>"pool_secondary_id - Memory Pool Secondary Identifier " on page 204</p> <p>"pool_config_size - Configured Size of Memory Pool " on page 205</p> <p>"pool_watermark - Memory Pool Watermark " on page 206</p>
progress	<p>"progress_completed_units - Completed Progress Work Units " on page 227</p> <p>"progress_description - Progress Description " on page 226</p> <p>"progress_seq_num - Progress Sequence Number " on page 225</p> <p>"progress_start_time - Progress Start Time " on page 226</p> <p>"progress_total_units - Total Progress Work Units " on page 227</p> <p>"progress_work_metric - Progress Work Metric " on page 226</p>
progress_list	<p>"progress_list_cur_seq_num - Current Progress List Sequence Number " on page 225</p> <p>"progress_list_attr - Current Progress List Attributes " on page 225</p>
rollforward	<p>"node_number - Node Number " on page 181</p> <p>"rf_type - Rollforward Type " on page 326</p> <p>"rf_log_num - Log Being Rolled Forward " on page 326</p> <p>"rf_status - Log Phase " on page 326</p> <p>"rf_timestamp - Rollforward Timestamp " on page 325</p> <p>"ts_name - Tablespace Being Rolled Forward " on page 325</p>

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
stmt	"agents_top - Number of Agents Created " on page 412
	"blocking_cursor - Blocking Cursor " on page 471
	"consistency_token - Package Consistency Token " on page 388
	"creator - Application Creator " on page 390
	"cursor_name - Cursor Name " on page 389
	"degree_parallelism - Degree of Parallelism " on page 412
	"fetch_count - Number of Successful Fetches " on page 394
	"int_rows_deleted - Internal Rows Deleted " on page 358
	"int_rows_inserted - Internal Rows Inserted " on page 359
	"int_rows_updated - Internal Rows Updated " on page 358
	"num_agents - Number of Agents Working on a Statement " on page 411
	"package_name - Package Name " on page 387
	"package_version_id - Package Version " on page 388
	"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads " on page 265
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads " on page 266
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads " on page 262
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads " on page 263
	"query_card_estimate - Query Number of Rows Estimate " on page 395
	"query_cost_estimate - Query Cost Estimate " on page 396
	"rows_read - Rows Read " on page 356
	"rows_written - Rows Written " on page 355
	"section_number - Section Number " on page 389
	"sort_overflows - Sort Overflows " on page 211
	"stmt_elapsed_time - Most Recent Statement Elapsed Time " on page 392
	"stmt_node_number - Statement Node " on page 383
	"stmt_operation/operation - Statement Operation " on page 386
	"stmt_sorts - Statement Sorts " on page 394
	"stmt_start - Statement Operation Start Timestamp " on page 391
	"stmt_stop - Statement Operation Stop Timestamp " on page 391
	"stmt_sys_cpu_time - System CPU Time used by Statement " on page 415
	"stmt_text - SQL Dynamic Statement Text " on page 393
	"stmt_type - Statement Type " on page 385
	"stmt_usr_cpu_time - User CPU Time used by Statement " on page 414
"total_sort_time - Total Sort Time " on page 210	

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
stmt_transmissions	<p>“elapsed_exec_time - Statement Execution Elapsed Time ” on page 467</p> <p>“host_response_time - Host Response Time ” on page 468</p> <p>“max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes ” on page 453</p> <p>“max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes ” on page 454</p> <p>“max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes ” on page 455</p> <p>“max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes ” on page 456</p> <p>“max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes ” on page 457</p> <p>“max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes ” on page 458</p> <p>“max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes ” on page 459</p> <p>“max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes ” on page 460</p> <p>“max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes ” on page 461</p> <p>“max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes ” on page 462</p> <p>“max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes ” on page 462</p> <p>“max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes ” on page 453</p> <p>“max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes ” on page 453</p> <p>“max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes ” on page 454</p> <p>“max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes ” on page 455</p> <p>“max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes ” on page 456</p> <p>“max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes ” on page 457</p> <p>“max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes ” on page 458</p> <p>“max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes ” on page 459</p> <p>“max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes ” on page 460</p> <p>“max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes ” on page 461</p> <p>“max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes ” on page 462</p>

System Monitor Logical data groups

Table 14. 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 463
	"max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms " on page 463
	"max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms " on page 464
	"max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms " on page 465
	"max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms " on page 465
	"max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms " on page 466
	"network_time_top - Maximum Network Time for Statement " on page 466
	"network_time_bottom - Minimum Network Time for Statement " on page 467
	"outbound_bytes_received - Outbound Number of Bytes Received " on page 450
	"outbound_bytes_sent - Outbound Number of Bytes Sent " on page 449
	"outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent " on page 451
	"outbound_bytes_received_top - Maximum Outbound Number of Bytes Received " on page 451
	"outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent " on page 452
	"outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received " on page 452
	"sql_chains - Number of SQL Chains Attempted " on page 444
	"sql_stmts - Number of SQL Statements Attempted " on page 444
	subsection
"rows_written - Rows Written " on page 355	
"ss_exec_time - Subsection Execution Elapsed Time " on page 405	
"ss_node_number - Subsection Node Number " on page 404	
"ss_number - Subsection Number " on page 404	
"ss_status - Subsection Status " on page 404	
"ss_sys_cpu_time - System CPU Time used by Subsection " on page 418	
"ss_usr_cpu_time - User CPU Time used by Subsection " on page 417	
"tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed " on page 407	
"tq_id_waiting_on - Waited on Node on a Tablequeue " on page 409	
"tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows " on page 408	
"tq_node_waited_for - Waited for Node on a Tablequeue " on page 406	
"tq_rows_read - Number of Rows Read from Tablequeues " on page 407	
"tq_rows_written - Number of Rows Written to Tablequeues " on page 408	
"tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed " on page 406	
"tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue " on page 405	

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
table	"data_object_pages - Data Object Pages " on page 361
	"data_partition_id - Data Partition Identifier monitor element" on page 188
	"index_object_pages - Index Object Pages " on page 362
	"lob_object_pages - LOB Object Pages " on page 362
	"long_object_pages - Long Object Pages " on page 363
	"overflow_accesses - Accesses to Overflowed Records " on page 357
	"page_reorgs - Page Reorganizations " on page 360
	"rows_read - Rows Read " on page 356
	"rows_written - Rows Written " on page 355
	"table_file_id - Table File ID " on page 360
	"table_name - Table Name " on page 351
	"table_schema - Table Schema Name " on page 352
	"tablespace_id - Table Space Identification " on page 328
	"data_partition_id - Data Partition Identifier monitor element" on page 188
	"table_type - Table Type " on page 350
"xda_object_pages - XDA Object Pages " on page 267	
table_list	"db_conn_time - Database Activation Timestamp " on page 155
	"db_name - Database Name " on page 154
	"db_path - Database Path " on page 155
	"input_db_alias - Input Database Alias " on page 420
	"last_reset - Last Reset Timestamp " on page 419
table_reorg	"data_partition_id - Data Partition Identifier monitor element" on page 188
	"reorg_completion - Reorganization Completion Flag " on page 366
	"reorg_current_counter - Reorganize Progress " on page 366
	"reorg_end - Table Reorganize End Time " on page 367
	"reorg_index_id - Index Used to Reorganize the Table " on page 367
	"reorg_max_counter - Total Amount of Reorganization " on page 366
	"reorg_max_phase - Maximum Reorganize Phase " on page 365
	"reorg_phase - Reorganize Phase " on page 365
	"reorg_phase_start - Reorganize Phase Start Time " on page 365
	"reorg_start - Table Reorganize Start Time " on page 367
	"reorg_status - Table Reorganize Status " on page 364
	"reorg_tbsp_id - Table Space Where Table or Data partition is Reorganized " on page 367
	"reorg_type - Table Reorganize Attributes " on page 363
"reorg_rows_compressed - Rows Compressed " on page 368	
"reorg_rows_rejected_for_compression - Rows Rejected for Compression " on page 368	

System Monitor Logical data groups

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
tablespace	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_read_time - Direct Read Time " on page 272
	"direct_reads - Direct Reads From Database " on page 269
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_write_time - Direct Write Time " on page 272
	"direct_writes - Direct Writes to Database " on page 269
	"files_closed - Database Files Closed " on page 244
	"fs_caching - File System Caching " on page 350
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests " on page 250
	"pool_async_data_reads - Buffer Pool Asynchronous Data Reads " on page 245
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes " on page 246
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests " on page 251
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads " on page 248
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes " on page 247
	"pool_async_read_time - Buffer Pool Asynchronous Read Time " on page 249
	"pool_async_write_time - Buffer Pool Asynchronous Write Time " on page 249
	"pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests " on page 259
	"pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads " on page 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 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_data_writes - Buffer Pool Data Writes " on page 237
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_index_writes - Buffer Pool Index Writes " on page 241
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers " on page 254
	"pool_read_time - Total Buffer Pool Physical Read Time " on page 243

Table 14. 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 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads " on page 265
	"pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads " on page 266
	"pool_write_time - Total Buffer Pool Physical Write Time " on page 243
	"pool_xda_l_reads - Buffer Pool XDA Data Logical Reads " on page 262
	"pool_xda_p_reads - Buffer Pool XDA Data Physical Reads " on page 263
	"pool_xda_writes - Buffer Pool XDA Data Writes " on page 264
	"tablespace_auto_resize_enabled - Auto-resize enabled " on page 335
	"tablespace_content_type - Table Space Contents Type " on page 330
	"tablespace_cur_pool_id - Buffer Pool Currently Being Used " on page 332
	"tablespace_extent_size - Table Space Extent Size " on page 331
	"tablespace_id - Table Space Identification " on page 328
	"tablespace_name - Table Space Name " on page 328
	"tablespace_next_pool_id - Buffer Pool That Will Be Used at Next Startup " on page 332
	"tablespace_page_size - Table Space Page Size " on page 331
	"tablespace_prefetch_size - Table Space Prefetch Size " on page 331
	"tablespace_rebalancer_mode - Rebalancer Mode " on page 338
"tablespace_type - Table Space Type " on page 329	
"tablespace_using_auto_storage - Using automatic storage " on page 334	
tablespace_container	"container_accessible - Accessibility of Container " on page 346
	"container_id - Container Identification " on page 344
	"container_name - Container Name " on page 344
	"container_stripe_set - Stripe Set " on page 345
	"container_total_pages - Total Pages in Container " on page 345
	"container_type - Container Type " on page 344
"container_usable_pages - Usable Pages in Container " on page 345	
tablespace_list	"db_conn_time - Database Activation Timestamp " on page 155
	"db_name - Database Name " on page 154
	"db_path - Database Path " on page 155
	"input_db_alias - Input Database Alias " on page 420
	"last_reset - Last Reset Timestamp " on page 419

System Monitor Logical data groups

Table 14. 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 335
	"tablespace_free_pages - Free Pages in Table Space " on page 334
	"tablespace_increase_size - Increase size in bytes " on page 336
	"tablespace_increase_size_percent - Increase size by percent " on page 337
	"tablespace_initial_size - Initial table space size " on page 335
	"tablespace_last_resize_failed - Last resize attempt failed " on page 337
	"tablespace_last_resize_time - Time of last successful resize " on page 337
	"tablespace_max_size - Maximum table space size " on page 336
	"tablespace_min_recovery_time - Minimum Recovery Time For Rollforward " on page 343
	"tablespace_num_containers - Number of Containers in Table Space " on page 343
	"tablespace_num_quiescers - Number of Quiescers " on page 340
	"tablespace_num_ranges - Number of Ranges in the Table Space Map " on page 346
	"tablespace_page_top - Table Space High Water Mark " on page 334
	"tablespace_pending_free_pages - Pending Free Pages in Table Space " on page 334
	"tablespace_prefetch_size - Table Space Prefetch Size " on page 331
	"tablespace_rebalancer_extents_processed - Number of Extents the Rebalancer has Processed " on page 339
	"tablespace_rebalancer_extents_remaining - Total Number of Extents to be Processed by the Rebalancer " on page 339
	"tablespace_rebalancer_last_extent_moved - Last Extent Moved by the Rebalancer " on page 340
	"tablespace_rebalancer_priority - Current Rebalancer Priority " on page 340
	"tablespace_rebalancer_restart_time - Rebalancer Restart Time " on page 339
	"tablespace_rebalancer_start_time - Rebalancer Start Time " on page 338
	"tablespace_state - Table Space State " on page 330
	"tablespace_state_change_object_id - State Change Object Identification " on page 342
	"tablespace_state_change_ts_id - State Change Table Space Identification " on page 343
	"tablespace_total_pages - Total Pages in Table Space " on page 332
	"tablespace_usable_pages - Usable Pages in Table Space " on page 333
	"tablespace_used_pages - Used Pages in Table Space " on page 333
tablespace_quiescer	"quiescer_agent_id - Quiescer Agent Identification " on page 341
	"quiescer_auth_id - Quiescer User Authorization Identification " on page 341
	"quiescer_obj_id - Quiescer Object Identification " on page 342
	"quiescer_state - Quiescer State " on page 342
	"quiescer_ts_id - Quiescer Table Space Identification " on page 341
tablespace_range	"range_adjustment - Range Adjustment " on page 348
	"range_container_id - Range Container " on page 349
	"range_end_stripe - End Stripe " on page 348
	"range_max_extent - Maximum Extent in Range " on page 348
	"range_max_page_number - Maximum Page in Range " on page 347
	"range_num_containers - Number of Containers in Range " on page 349
	"range_number - Range Number " on page 347
	"range_offset - Range Offset " on page 349
	"range_start_stripe - Start Stripe " on page 348
"range_stripe_set_number - Stripe Set Number " on page 347	

Table 14. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element
utility_info	"utility_dbname - Database Operated on by Utility " on page 223 "utility_id - Utility ID " on page 223 "utility_invoker_type - Utility Invoker Type " on page 229 "utility_state - Utility State " on page 228 "utility_type - Utility Type " on page 223 "utility_priority - Utility Priority " on page 224 "utility_start_time - Utility Start Time " on page 224 "utility_description - Utility Description " on page 224 "node_number - Node Number " on page 181

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 15. 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.

System Monitor Logical data groups

Prolog:

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

Table 16. 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 17. 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.
Deadlocked Connection Event	event_dlconn	One for each connection involved in the deadlock, includes applications involved and locks in contention.
Deadlocked Connection Event with Details	event_detailed_dlconn, lock	One for each connection involved in the deadlock, includes applications involved, locks in contention, current statement information, and other locks held by the application contention.
Overflow	event_overflow	Number of records lost - generated when writer cannot keep up with a (non-blocked) event monitor.

Table 17. Event Monitor Data Stream: Contents Section (continued)

Event type	Logical data group	Available information
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.

Epilog:

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

Table 18. Event Monitor Data Stream: Epilog Section

Event type	Logical data group	Available information
Database Event	event_db	Database manager level data.
Buffer Pool Event	event_bufferpool	Buffer pool level data.
Table Space Event	event_tablespace	Table space level data.
Table Event	event_table	Table level data.

Related concepts:

- “Database system monitor data organization” on page 3
- “Event monitors” on page 59

Related tasks:

- “Collecting information about database system events” on page 61

Related reference:

- “Event monitor sample output” on page 80

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.

System Monitor Logical data groups

Table 19. Event Monitor Logical Data Groups and Monitor Elements

Event logical data groups	Monitor element name
event_bufferpool	"bp_name - Buffer Pool Name " on page 255
	"bp_id - Buffer Pool ID " on page 232
	"db_name - Database Name " on page 154
	"db_path - Database Path " on page 155
	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_read_time - Direct Read Time " on page 272
	"direct_reads - Direct Reads From Database " on page 269
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_write_time - Direct Write Time " on page 272
	"direct_writes - Direct Writes to Database " on page 269
	"event_time - Event Time " on page 424
	"evmon_activates - Number of Event Monitor Activations " on page 425
	"evmon_flushes - Number of Event Monitor Flushes " on page 424
	"files_closed - Database Files Closed " on page 244
	"partial_record - Partial Record " on page 423
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests " on page 250
	"pool_async_data_reads - Buffer Pool Asynchronous Data Reads " on page 245
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes " on page 246
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads " on page 248
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes " on page 247
	"pool_async_read_time - Buffer Pool Asynchronous Read Time " on page 249
	"pool_async_write_time - Buffer Pool Asynchronous Write Time " on page 249
	"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_data_writes - Buffer Pool Data Writes " on page 237
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_index_writes - Buffer Pool Index Writes " on page 241
	"pool_read_time - Total Buffer Pool Physical Read Time " on page 243
	"pool_write_time - Total Buffer Pool Physical Write Time " on page 243

Table 19. 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 371
	"agent_id - Application Handle (agent ID) " on page 163
	"appl_id - Application ID " on page 169
	"appl_priority - Application Agent Priority " on page 179
	"appl_priority_type - Application Priority Type " on page 180
	"appl_section_inserts - Section Inserts monitor element" on page 281
	"appl_section_lookups - Section Lookups " on page 280
	"authority_lvl - User Authorization Level " on page 180
	"binds_precompiles - Binds/Precompiles Attempted " on page 383
	"cat_cache_inserts - Catalog Cache Inserts " on page 274
	"cat_cache_lookups - Catalog Cache Lookups " on page 273
	"cat_cache_overflows - Catalog Cache Overflows " on page 275
	"commit_sql_stmts - Commit Statements Attempted " on page 375
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements " on page 378
	"deadlocks - Deadlocks Detected " on page 303
	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_read_time - Direct Read Time " on page 272
	"direct_reads - Direct Reads From Database " on page 269
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_write_time - Direct Write Time " on page 272
	"direct_writes - Direct Writes to Database " on page 269
	"disconn_time - Database Deactivation Timestamp " on page 156
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted " on page 373
	"failed_sql_stmts - Failed Statement Operations " on page 374
	"hash_join_overflows - Hash Join Overflows " on page 218
	"hash_join_small_overflows - Hash Join Small Overflows " on page 218
	"int_auto_rebinds - Internal Automatic Rebinds " on page 379
	"int_commits - Internal Commits " on page 380
	"int_deadlock_rollback - Internal Rollbacks Due To Deadlock " on page 382
	"int_rollback - Internal Rollbacks " on page 381
	"int_rows_deleted - Internal Rows Deleted " on page 358
	"int_rows_inserted - Internal Rows Inserted " on page 359
	"int_rows_updated - Internal Rows Updated " on page 358
	"lock_escalation - Lock Escalation " on page 311
	"lock_timeouts - Number of Lock Timeouts " on page 309
	"lock_wait_time - Time Waited On Locks " on page 319
	"lock_waits - Lock Waits " on page 318
	"partial_record - Partial Record " on page 423
	"pkg_cache_inserts - Package Cache Inserts " on page 279
	"pkg_cache_lookups - Package Cache Lookups " on page 277
	"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_data_writes - Buffer Pool Data Writes " on page 237

System Monitor Logical data groups

Table 19. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name
event_conn (continued)	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_index_writes - Buffer Pool Index Writes " on page 241
	"pool_read_time - Total Buffer Pool Physical Read Time " on page 243
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"pool_write_time - Total Buffer Pool Physical Write Time " on page 243
	"prefetch_wait_time - Time Waited for Prefetch " on page 256
	"priv_workspace_num_overflows - Private Workspace Overflows " on page 285
	"priv_workspace_section_inserts - Private Workspace Section Inserts " on page 287
	"priv_workspace_section_lookups - Private Workspace Section Lookups " on page 286
	"priv_workspace_size_top - Maximum Private Workspace Size " on page 285
	"rej_curs_blk - Rejected Block Cursor Requests " on page 370
	"rollback_sql_stmts - Rollback Statements Attempted " on page 376
	"rows_read - Rows Read " on page 356
	"rows_selected - Rows Selected " on page 355
	"int_rows_updated - Internal Rows Updated " on page 358
	"rows_written - Rows Written " on page 355
	"select_sql_stmts - Select SQL Statements Executed " on page 377
	"sequence_no - Sequence Number " on page 172
	"shr_workspace_num_overflows - Shared Workspace Overflows " on page 283
	"shr_workspace_section_inserts - Shared Workspace Section Inserts " on page 284
	"shr_workspace_section_lookups - Shared Workspace Section Lookups " on page 283
	"shr_workspace_size_top - Maximum Shared Workspace Size " on page 282
	"sort_overflows - Sort Overflows " on page 211
	"static_sql_stmts - Static SQL Statements Attempted " on page 373
	"system_cpu_time - System CPU Time " on page 416
	"total_hash_joins - Total Hash Joins " on page 215
	"total_hash_loops - Total Hash Loops " on page 217
	"total_sec_cons - Secondary Connections " on page 200
	"total_sort_time - Total Sort Time " on page 210
	"total_sorts - Total Sorts " on page 210
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed " on page 378
	"unread_prefetch_pages - Unread Prefetch Pages " on page 256
	"user_cpu_time - User CPU Time " on page 416
	"x_lock_escals - Exclusive Lock Escalations " on page 305
	"xquery_stmts - XQuery Statements Attempted" on page 383

Table 19. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name
event_connheader	"agent_id - Application Handle (agent ID) " on page 163
	"appl_id - Application ID " on page 169
	"appl_name - Application Name " on page 168
	"auth_id - Authorization ID " on page 172
	"client_db_alias - Database Alias Used by Application " on page 174
	"client_nname - Configuration NNAME of Client " on page 173
	"client_pid - Client Process ID " on page 177
	"client_platform - Client Operating Platform " on page 178
	"client_prdid - Client Product/Version ID " on page 174
	"client_protocol - Client Communication Protocol " on page 178
	"codepage_id - ID of Code Page Used by Application " on page 166
	"conn_time - Time of Database Connection " on page 156
	"corr_token - DRDA Correlation Token " on page 177
	"execution_id - User Login ID " on page 176
	"node_number - Node Number " on page 181
	"sequence_no - Sequence Number " on page 172
"territory_code - Database Territory Code " on page 179	
event_connmemuse	"node_number - Node Number " on page 181
	"pool_cur_size - Current Size of Memory Pool " on page 204
	"pool_id - Memory Pool Identifier " on page 203
	"pool_secondary_id - Memory Pool Secondary Identifier " on page 204
	"pool_config_size - Configured Size of Memory Pool " on page 205
"pool_watermark - Memory Pool Watermark " on page 206	
event_data_value	"evmon_activates - Number of Event Monitor Activations " on page 425
	"deadlock_id - Deadlock Event Identifier " on page 312
	"deadlock_node - Partition Number Where Deadlock Occurred " on page 312
	"participant_no - Participant within Deadlock " on page 313
	"stmt_value_type - Value type " on page 401
	"stmt_history_id - Statement history identifier " on page 397
	"stmt_value_isnull - Value has null value " on page 401
	"stmt_value_data - Value data " on page 402
	"stmt_value_index - Value index " on page 402
"stmt_value_isreoptvalue - Variable used for statement reoptimization " on page 403	

System Monitor Logical data groups

Table 19. 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 215
	"appl_section_inserts - Section Inserts monitor element" on page 281
	"appl_section_lookups - Section Lookups " on page 280
	"binds_precompiles - Binds/Precompiles Attempted " on page 383
	"cat_cache_inserts - Catalog Cache Inserts " on page 274
	"cat_cache_lookups - Catalog Cache Lookups " on page 273
	"cat_cache_overflows - Catalog Cache Overflows " on page 275
	"cat_cache_size_top - Catalog Cache High Water Mark " on page 276
	"catalog_node - Catalog Node Number " on page 158
	"catalog_node_name - Catalog Node Network Name " on page 157
	"commit_sql_stmts - Commit Statements Attempted " on page 375
	"connections_top - Maximum Number of Concurrent Connections " on page 183
	"db_heap_top - Maximum Database Heap Allocated " on page 288
	"ddl_sql_stmts - Data Definition Language (DDL) SQL Statements " on page 378
	"deadlocks - Deadlocks Detected " on page 303
	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_read_time - Direct Read Time " on page 272
	"direct_reads - Direct Reads From Database " on page 269
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_write_time - Direct Write Time " on page 272
	"direct_writes - Direct Writes to Database " on page 269
	"disconn_time - Database Deactivation Timestamp " on page 156
	"dynamic_sql_stmts - Dynamic SQL Statements Attempted " on page 373
	"evmon_activates - Number of Event Monitor Activations " on page 425
	"evmon_flushes - Number of Event Monitor Flushes " on page 424
	"failed_sql_stmts - Failed Statement Operations " on page 374
	"files_closed - Database Files Closed " on page 244
	"hash_join_overflows - Hash Join Overflows " on page 218
	"hash_join_small_overflows - Hash Join Small Overflows " on page 218
	"int_auto_rebinds - Internal Automatic Rebinds " on page 379
	"int_commits - Internal Commits " on page 380
	"int_rollbacks - Internal Rollbacks " on page 381
	"int_rows_deleted - Internal Rows Deleted " on page 358
	"int_rows_inserted - Internal Rows Inserted " on page 359
	"int_rows_updated - Internal Rows Updated " on page 358
	"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 319
	"lock_waits - Lock Waits " on page 318
	"log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages " on page 294
	"log_read_time - Log Read Time " on page 296
	"log_reads - Number of Log Pages Read " on page 291
	"log_to_redo_for_recovery - Amount of Log to be Redone for Recovery " on page 295
	"log_write_time - Log Write Time " on page 295
	"log_writes - Number of Log Pages Written " on page 291
	"num_log_read_io - Number of Log Reads " on page 297
	"num_log_write_io - Number of Log Writes " on page 296

Table 19. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name
event_db (continued)	"partial_record - Partial Record " on page 423
	"pkg_cache_inserts - Package Cache Inserts " on page 279
	"pkg_cache_lookups - Package Cache Lookups " on page 277
	"pkg_cache_num_overflows - Package Cache Overflows " on page 279
	"pkg_cache_size_top - Package Cache High Water Mark " on page 280
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests " on page 250
	"pool_async_data_reads - Buffer Pool Asynchronous Data Reads " on page 245
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes " on page 246
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests " on page 251
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads " on page 248
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes " on page 247
	"pool_async_read_time - Buffer Pool Asynchronous Read Time " on page 249
	"pool_async_write_time - Buffer Pool Asynchronous Write Time " on page 249
	"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_data_writes - Buffer Pool Data Writes " on page 237
	"pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered " on page 252
	"pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered " on page 254
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_index_writes - Buffer Pool Index Writes " on page 241
	"pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered " on page 251
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers " on page 254
	"pool_read_time - Total Buffer Pool Physical Read Time " on page 243
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"pool_write_time - Total Buffer Pool Physical Write Time " on page 243
	"post_shrthreshold_hash_joins - Post threshold hash joins " on page 216
	"post_shrthreshold_sorts - Post threshold sorts " on page 214
	"prefetch_wait_time - Time Waited for Prefetch " on page 256
	"priv_workspace_num_overflows - Private Workspace Overflows " on page 285
	"priv_workspace_section_inserts - Private Workspace Section Inserts " on page 287
	"priv_workspace_section_lookups - Private Workspace Section Lookups " on page 286
	"priv_workspace_size_top - Maximum Private Workspace Size " on page 285
	"rollback_sql_stmts - Rollback Statements Attempted " on page 376
	"rows_deleted - Rows Deleted " on page 353
	"rows_inserted - Rows Inserted " on page 353
	"rows_read - Rows Read " on page 356
	"rows_selected - Rows Selected " on page 355
	"rows_updated - Rows Updated " on page 354

System Monitor Logical data groups

Table 19. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name
event_db (continued)	"sec_log_used_top - Maximum Secondary Log Space Used " on page 289
	"select_sql_stmts - Select SQL Statements Executed " on page 377
	"server_platform - Server Operating System " on page 152
	"shr_workspace_num_overflows - Shared Workspace Overflows " on page 283
	"shr_workspace_section_inserts - Shared Workspace Section Inserts " on page 284
	"shr_workspace_section_lookups - Shared Workspace Section Lookups " on page 283
	"shr_workspace_size_top - Maximum Shared Workspace Size " on page 282
	"sort_overflows - Sort Overflows " on page 211
	"static_sql_stmts - Static SQL Statements Attempted " on page 373
	"tot_log_used_top - Maximum Total Log Space Used " on page 290
	"total_cons - Connects Since Database Activation " on page 194
	"total_hash_joins - Total Hash Joins " on page 215
	"total_hash_loops - Total Hash Loops " on page 217
	"total_sort_time - Total Sort Time " on page 210
	"total_sorts - Total Sorts " on page 210
	"uid_sql_stmts - Update/Insert/Delete SQL Statements Executed " on page 378
	"unread_prefetch_pages - Unread Prefetch Pages " on page 256
"x_lock_escals - Exclusive Lock Escalations " on page 305	
"xquery_stmts - XQuery Statements Attempted" on page 383	
event_dbheader	"conn_time - Time of Database Connection " on page 156
	"db_name - Database Name " on page 154
	"db_path - Database Path " on page 155
event_dbmemuse	"node_number - Node Number " on page 181
	"pool_cur_size - Current Size of Memory Pool " on page 204
	"pool_id - Memory Pool Identifier " on page 203
	"pool_config_size - Configured Size of Memory Pool " on page 205
	"pool_watermark - Memory Pool Watermark " on page 206
event_deadlock	"deadlock_id - Deadlock Event Identifier " on page 312
	"deadlock_node - Partition Number Where Deadlock Occurred " on page 312
	"dl_conns - Connections Involved in Deadlock " on page 311
	"evmon_activates - Number of Event Monitor Activations " on page 425
	"rolled_back_agent_id - Rolled Back Agent " on page 323
	"rolled_back_appl_id - Rolled Back Application " on page 323
	"rolled_back_participant_no - Rolled Back Application Participant " on page 313
	"rolled_back_sequence_no - Rolled Back Sequence Number " on page 324
"start_time - Event Start Time " on page 392	

Table 19. 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 163
	"appl_id - Application ID " on page 169
	"appl_id_holding_lk - Application ID Holding Lock " on page 322
	"blocking_cursor - Blocking Cursor " on page 471
	"consistency_token - Package Consistency Token " on page 388
	"creator - Application Creator " on page 390
	"cursor_name - Cursor Name " on page 389
	"deadlock_id - Deadlock Event Identifier " on page 312
	"deadlock_node - Partition Number Where Deadlock Occurred " on page 312
	"data_partition_id - Data Partition Identifier monitor element" on page 188
	"evmon_activates - Number of Event Monitor Activations " on page 425
	"lock_escalation - Lock Escalation " on page 311
	"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 308
	"lock_wait_start_time - Lock Wait Start Timestamp " on page 320
	"locks_held - Locks Held " on page 302
	"locks_in_list - Number of Locks Reported " on page 314
	"package_name - Package Name " on page 387
	"package_version_id - Package Version " on page 388
	"participant_no - Participant within Deadlock " on page 313
	"participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application " on page 313
	"section_number - Section Number " on page 389
	"sequence_no - Sequence Number " on page 172
	"sequence_no_holding_lk - Sequence Number Holding Lock " on page 323
	"start_time - Event Start Time " on page 392
	"stmt_operation/operation - Statement Operation " on page 386
	"stmt_text - SQL Dynamic Statement Text " on page 393
	"stmt_type - Statement Type " on page 385
	"table_name - Table Name " on page 351
	"table_schema - Table Schema Name " on page 352
	"tablespace_name - Table Space Name " on page 328

System Monitor Logical data groups

Table 19. 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 163
	"appl_id - Application ID " on page 169
	"appl_id_holding_lk - Application ID Holding Lock " on page 322
	"deadlock_id - Deadlock Event Identifier " on page 312
	"data_partition_id - Data Partition Identifier monitor element" on page 188
	"deadlock_node - Partition Number Where Deadlock Occurred " on page 312
	"evmon_activates - Number of Event Monitor Activations " on page 425
	"lock_attributes - Lock Attributes " on page 314
	"lock_count - Lock Count " on page 316
	"lock_current_mode - Original Lock Mode Before Conversion " on page 316
	"lock_escalation - Lock Escalation " on page 311
	"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 314
	"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 308
	"lock_release_flags - Lock Release Flags " on page 315
	"lock_wait_start_time - Lock Wait Start Timestamp " on page 320
	"participant_no - Participant within Deadlock " on page 313
	"participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application " on page 313
	"sequence_no - Sequence Number " on page 172
	"sequence_no_holding_lk - Sequence Number Holding Lock " on page 323
	"start_time - Event Start Time " on page 392
	"table_name - Table Name " on page 351
	"table_schema - Table Schema Name " on page 352
"tablespace_name - Table Space Name " on page 328	
event_log_header	"byte_order - Byte Order of Event Data " on page 422
	"codepage_id - ID of Code Page Used by Application " on page 166
	"event_monitor_name - Event Monitor Name " on page 423
	"num_nodes_in_db2_instance - Number of Nodes in Partition " on page 420
	"server_prdid - Server Product/Version ID " on page 150
	"server_instance_name - Server Instance Name " on page 149
	"territory_code - Database Territory Code " on page 179
"version - Version of Monitor Data " on page 423	
event_overflow	"count - Number of Event Monitor Overflows " on page 421
	"first_overflow_time - Time of First Event Overflow " on page 422
	"last_overflow_time - Time of Last Event Overflow " on page 422
	"node_number - Node Number " on page 181
event_start	"start_time - Event Start Time " on page 392

Table 19. 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 163
	"agents_top - Number of Agents Created " on page 412
	"appl_id - Application ID " on page 169
	"blocking_cursor - Blocking Cursor " on page 471
	"consistency_token - Package Consistency Token " on page 388
	"creator - Application Creator " on page 390
	"cursor_name - Cursor Name " on page 389
	"fetch_count - Number of Successful Fetches " on page 394
	"int_rows_deleted - Internal Rows Deleted " on page 358
	"int_rows_inserted - Internal Rows Inserted " on page 359
	"int_rows_updated - Internal Rows Updated " on page 358
	"package_name - Package Name " on page 387
	"package_version_id - Package Version " on page 388
	"partial_record - Partial Record " on page 423
	"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"rows_read - Rows Read " on page 356
	"rows_written - Rows Written " on page 355
	"section_number - Section Number " on page 389
	"sequence_no - Sequence Number " on page 172
	"sort_overflows - Sort Overflows " on page 211
	"sql_req_id - Request Identifier for SQL Statement " on page 425
	"sqlca - SQL Communications Area (SQLCA) " on page 395
	"start_time - Event Start Time " on page 392
	"stmt_operation/operation - Statement Operation " on page 386
	"stmt_text - SQL Dynamic Statement Text " on page 393
	"stmt_type - Statement Type " on page 385
	"stop_time - Event Stop Time " on page 391
	"system_cpu_time - System CPU Time " on page 416
	"total_sort_time - Total Sort Time " on page 210
	"total_sorts - Total Sorts " on page 210
	"user_cpu_time - User CPU Time " on page 416

System Monitor Logical data groups

Table 19. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name
event_stmt_history	"evmon_activates - Number of Event Monitor Activations " on page 425
	"deadlock_id - Deadlock Event Identifier " on page 312
	"deadlock_node - Partition Number Where Deadlock Occurred " on page 312
	"participant_no - Participant within Deadlock " on page 313
	"stmt_history_id - Statement history identifier " on page 397
	"stmt_first_use_time - Statement first use time " on page 397
	"stmt_last_use_time - Statement last use time monitor element" on page 398
	"stmt_lock_timeout - Statement lock timeout " on page 398
	"stmt_isolation - Statement isolation " on page 398
	"package_name - Package Name " on page 387
	"package_version_id - Package Version " on page 388
	"section_number - Section Number " on page 389
	"creator - Application Creator " on page 390
	"stmt_type - Statement Type " on page 385
	"stmt_text - SQL Dynamic Statement Text " on page 393
	"comp_env_desc - Compilation environment handle " on page 401
	"stmt_nest_level - Statement nesting level " on page 399
	"stmt_invocation_id - Statement invocation identifier " on page 399
	"stmt_query_id - Statement query identifier " on page 400
	"stmt_source_id - Statement source identifier " on page 400
"sequence_no - Sequence Number " on page 172	
"stmt_pkgcache_id - Statement package cache identifier " on page 400	
event_subsection	"agent_id - Application Handle (agent ID) " on page 163
	"num_agents - Number of Agents Working on a Statement " on page 411
	"partial_record - Partial Record " on page 423
	"ss_exec_time - Subsection Execution Elapsed Time " on page 405
	"ss_node_number - Subsection Node Number " on page 404
	"ss_number - Subsection Number " on page 404
	"ss_sys_cpu_time - System CPU Time used by Subsection " on page 418
	"ss_usr_cpu_time - User CPU Time used by Subsection " on page 417
	"tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows " on page 408
	"tq_rows_read - Number of Rows Read from Tablequeues " on page 407
	"tq_rows_written - Number of Rows Written to Tablequeues " on page 408
	"tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed " on page 406

Table 19. 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 361</p> <p>“event_time - Event Time ” on page 424</p> <p>“evmon_activates - Number of Event Monitor Activations ” on page 425</p> <p>“evmon_flushes - Number of Event Monitor Flushes ” on page 424</p> <p>“index_object_pages - Index Object Pages ” on page 362</p> <p>“lob_object_pages - LOB Object Pages ” on page 362</p> <p>“long_object_pages - Long Object Pages ” on page 363</p> <p>“overflow_accesses - Accesses to Overflowed Records ” on page 357</p> <p>“page_reorgs - Page Reorganizations ” on page 360</p> <p>“partial_record - Partial Record ” on page 423</p> <p>“rows_read - Rows Read ” on page 356</p> <p>“rows_written - Rows Written ” on page 355</p> <p>“table_name - Table Name ” on page 351</p> <p>“table_schema - Table Schema Name ” on page 352</p> <p>“table_type - Table Type ” on page 350</p> <p>“data_partition_id - Data Partition Identifier monitor element” on page 188</p>

System Monitor Logical data groups

Table 19. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name
event_tablespace	"direct_read_reqs - Direct Read Requests " on page 270
	"direct_read_time - Direct Read Time " on page 272
	"direct_reads - Direct Reads From Database " on page 269
	"direct_write_reqs - Direct Write Requests " on page 271
	"direct_write_time - Direct Write Time " on page 272
	"direct_writes - Direct Writes to Database " on page 269
	"event_time - Event Time " on page 424
	"evmon_activates - Number of Event Monitor Activations " on page 425
	"evmon_flushes - Number of Event Monitor Flushes " on page 424
	"files_closed - Database Files Closed " on page 244
	"partial_record - Partial Record " on page 423
	"pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests " on page 250
	"pool_async_data_reads - Buffer Pool Asynchronous Data Reads " on page 245
	"pool_async_data_writes - Buffer Pool Asynchronous Data Writes " on page 246
	"pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests " on page 251
	"pool_async_index_reads - Buffer Pool Asynchronous Index Reads " on page 248
	"pool_async_index_writes - Buffer Pool Asynchronous Index Writes " on page 247
	"pool_async_read_time - Buffer Pool Asynchronous Read Time " on page 249
	"pool_async_write_time - Buffer Pool Asynchronous Write Time " on page 249
	"pool_data_l_reads - Buffer Pool Data Logical Reads " on page 233
	"pool_data_p_reads - Buffer Pool Data Physical Reads " on page 235
	"pool_data_writes - Buffer Pool Data Writes " on page 237
	"pool_index_l_reads - Buffer Pool Index Logical Reads " on page 238
	"pool_index_p_reads - Buffer Pool Index Physical Reads " on page 240
	"pool_index_writes - Buffer Pool Index Writes " on page 241
	"pool_no_victim_buffer - Buffer Pool No Victim Buffers " on page 254
	"pool_read_time - Total Buffer Pool Physical Read Time " on page 243
	"pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads " on page 234
	"pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads " on page 236
	"pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads " on page 239
	"pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads " on page 241
	"pool_write_time - Total Buffer Pool Physical Write Time " on page 243
	"tablespace_name - Table Space Name " on page 328

Table 19. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name
event_xact	"agent_id - Application Handle (agent ID) " on page 163
	"appl_id - Application ID " on page 169
	"lock_escals - Number of Lock Escalations " on page 304
	"lock_wait_time - Time Waited On Locks " on page 319
	"locks_held_top - Maximum Number of Locks Held " on page 310
	"partial_record - Partial Record " on page 423
	"prev_uow_stop_time - Previous Unit of Work Completion Timestamp " on page 184
	"rows_read - Rows Read " on page 356
	"rows_written - Rows Written " on page 355
	"sequence_no - Sequence Number " on page 172
	"system_cpu_time - System CPU Time " on page 416
	"uow_log_space_used - Unit of Work Log Space Used " on page 292
	"uow_start_time - Unit of Work Start Timestamp " on page 185
	"uow_status - Unit of Work Status " on page 187
	"uow_stop_time - Unit of Work Stop Timestamp " on page 185
	"user_cpu_time - User CPU Time " on page 416
"x_lock_escals - Exclusive Lock Escalations " on page 305	
lock	"lock_attributes - Lock Attributes " on page 314
	"lock_count - Lock Count " on page 316
	"lock_current_mode - Original Lock Mode Before Conversion " on page 316
	"lock_escalation - Lock Escalation " on page 311
	"lock_hold_count - Lock Hold Count " on page 316
	"lock_mode - Lock Mode " on page 306
	"lock_name - Lock Name " on page 314
	"lock_object_name - Lock Object Name " on page 308
	"lock_object_type - Lock Object Type Waited On " on page 308
	"lock_release_flags - Lock Release Flags " on page 315
	"lock_status - Lock Status " on page 307
	"node_number - Node Number " on page 181
	"table_file_id - Table File ID " on page 360
	"table_name - Table Name " on page 351
"table_schema - Table Schema Name " on page 352	
"tablespace_name - Table Space Name " on page 328	
"data_partition_id - Data Partition Identifier monitor element" on page 188	
sqlca	sqlcab
	sqlcode
	sqlerrml
	sqlcaid
	sqlerrmc
	sqlerrp
	sqlerrd
	sqlwarn
	sqlstate

System Monitor Logical data groups

Chapter 6. Monitor elements

Database system monitor elements

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.

Server identification and status monitor elements

- *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.

Description

A description of the data collected by 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

Server identification and status monitor elements

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

- db2start_time - Start Database Manager Timestamp monitor element
- server_nname - Configuration NNAME at Monitoring (Server) Database Partition monitor element
- server_instance_name - Server Instance Name monitor element
- server_db2_type - Database Manager Type at Monitored (Server) Node monitor element
- server_prdid - Server Product/Version ID monitor element
- server_version - Server Version monitor element
- service_level - Service Level monitor element
- server_platform - Server Operating System monitor element
- product_name - Product Name monitor element
- db2_status - Status of DB2 Instance monitor element
- time_zone_disp - Time Zone Displacement monitor element

db2start_time - Start Database Manager Timestamp

Element identifier db2start_time

Element type timestamp

Table 20. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The date and time that the database manager was started using the db2start command.

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.

Related reference:

- “time_stamp - Snapshot Time ” on page 420

server_name - Configuration NNAME at Monitoring (Server) Database Partition

Element identifier server_name
Element type information

Table 21. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Description

The name of the database partition being monitored by the database system monitor.

Usage This element can be used to identify the database server partition you are monitoring. 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 database server partitions. This database partition name is determined based on the *nname* configuration parameter.

This element only applies to Windows Environments where the NetBIOS LAN environment exists.

Related reference:

- “client_nname - Configuration NNAME of Client ” on page 173

server_instance_name - Server Instance Name

Element identifier server_instance_name
Element type information

Table 22. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Table 23. Event Monitoring Information

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

Description

The name of the database manager instance for which the snapshot was taken.

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. Along with *server_name*, this information can

Server identification and status monitor elements

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.

Related reference:

- “server_nname - Configuration NNAME at Monitoring (Server) Database Partition ” on page 149

server_db2_type - Database Manager Type at Monitored (Server) Node

Element identifier server_db2_type

Element type information

Table 24. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Description

Identifies the type of database manager being monitored.

Usage It contains one of the following types of configurations for the database manager:

API Symbolic Constant sqlf_nt_server	Command Line Processor Output 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*.

Related reference:

- “server_nname - Configuration NNAME at Monitoring (Server) Database Partition ” on page 149

server_prdid - Server Product/Version ID

Element identifier server_prdid

Element type information

Table 25. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Table 26. Event Monitoring Information

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

Description

The product and version that is running on the server.

Usage It is in the form PPPVRRM, where:
PPP is SQL

Server identification and status monitor elements

- 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-digit modification level

Related reference:

- “client_prdid - Client Product/Version ID ” on page 174

server_version - Server Version

Element identifier server_version

Element type information

Table 27. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Description

The version of the server returning the information.

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

Related reference:

- “server_prdid - Server Product/Version ID ” on page 150

service_level - Service Level

Element identifier service_level

Element type information

Table 28. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Server identification and status monitor elements

Description

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

server_platform - Server Operating System

Element identifier server_platform

Element type information

Table 29. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 30. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The operating system running the database server.

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*.

Related reference:

- “client_platform - Client Operating Platform ” on page 178
- “db_location - Database Location ” on page 157

product_name - Product Name

Element identifier product_name

Element type information

Table 31. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

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

db2_status - Status of DB2 Instance

Element identifier db2_status

Element type information

Table 32. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The current status of the instance of the database manager.

Usage You can use this element to determine the state of your database manager instance.

Server identification and status monitor elements

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.

Related reference:

- “db_status - Status of Database ” on page 156

time_zone_disp - Time Zone Displacement

Element identifier time_zone_disp

Element type information

Table 33. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Description

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

Usage All time reported by reported by the database system monitor is GMT, this displacement calculates the local time.

Database identification and status

Database identification and status monitor elements

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

- db_name - Database Name monitor element
- db_path - Database Path monitor element
- db_conn_time - Database Activation Timestamp monitor element
- conn_time - Time of Database Connection monitor element
- disconn_time - Database Deactivation Timestamp monitor element
- db_status - Status of Database monitor element
- catalog_node_name - Catalog Node Network Name monitor element
- db_location - Database Location monitor element

Database identification and status monitor elements

- catalog_node - Catalog Node Number monitor element
- last_backup - Last Backup Timestamp monitor element
- num_db_storage_paths - Number of automatic storage paths monitor element
num_db_storage_paths - Number of automatic storage paths monitor element
- db_storage_path - Automatic storage path monitor element
db_storage_path - Automatic storage path monitor element

db_name - Database Name

Element identifier	db_name
Element type	information

Table 34. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Database	dbase_remote	Basic
Application	appl_id_info	Basic
Application	appl_remote	Basic
Table Space	tablespace_list	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool
Table	table_list	Table
Lock	db_lock_list	Basic
Dynamic SQL	dynsql_list	Basic
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl_info	Basic

Table 35. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_dbheader	-

Description

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.

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 AS/400® and iSeries™ 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.

Related reference:

- “last_reset - Last Reset Timestamp ” on page 419
- “input_db_alias - Input Database Alias ” on page 420
- “client_db_alias - Database Alias Used by Application ” on page 174
- “db_path - Database Path ” on page 155

db_path - Database Path

Element identifier	db_path
Element type	information

Table 36. 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 37. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_dbheader	-

Description

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

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

Related reference:

- “input_db_alias - Input Database Alias ” on page 420
- “db_name - Database Name ” on page 154

db_conn_time - Database Activation Timestamp

Element identifier	db_conn_time
Element type	timestamp

Table 38. Snapshot Monitoring Information

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

Description

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.

Usage Use this element with the *disconn_time* monitor element to calculate the total connection time.

Related reference:

Database identification and status monitor elements

- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “time_stamp - Snapshot Time ” on page 420
- “conn_time - Time of Database Connection ” on page 156

conn_time - Time of Database Connection

Element identifier	conn_time
Element type	timestamp

Table 39. Event Monitoring Information

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

Description

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.

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).

Related reference:

- “db_conn_time - Database Activation Timestamp ” on page 155
- “disconn_time - Database Deactivation Timestamp ” on page 156

disconn_time - Database Deactivation Timestamp

Element identifier	disconn_time
Element type	timestamp

Table 40. Event Monitoring Information

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

Description

The date and time that the application disconnected from the database (at the database level, this is the time the last application disconnected).

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

Element identifier	db_status
Element type	information

Database identification and status monitor elements

Table 41. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The current status of the database.

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.

Related reference:

- “db2_status - Status of DB2 Instance ” on page 152

catalog_node_name - Catalog Node Network Name

Element identifier catalog_node_name

Element type information

Table 42. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 43. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The network name of the catalog node.

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

db_location - Database Location

Element identifier db_location

Element type information

Database identification and status monitor elements

Table 44. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The location of the database in relation to the application.

Usage Determine the relative location of the database server with respect to the application taking the snapshot. Values are:

- SQLM_LOCAL
- SQLM_REMOTE

Related reference:

- “server_platform - Server Operating System ” on page 152

catalog_node - Catalog Node Number

Element identifier catalog_node

Element type information

Table 45. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 46. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The node number of the node where the database catalog tables are stored.

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.

Related reference:

- “BACKUP DATABASE command” in *Command Reference*

last_backup - Last Backup Timestamp

Element identifier last_backup

Element type timestamp

Table 47. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Timestamp

Description

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

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.

num_db_storage_paths - Number of automatic storage paths

Element identifier	num_db_storage_paths
Element type	information

Table 48. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

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

Usage You can use this element with the db_storage_path monitor element to identify the storage paths that are associated with this database.

db_storage_path - Automatic storage path

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

Description

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.

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.

Related reference:

- “sto_path_free_sz - Automatic Storage Path Free Space ” on page 159
- “fs_id - Unique File System Identification Number ” on page 161
- “fs_total_size - Total Size of a File System ” on page 160
- “fs_type - File System Type ” on page 162
- “fs_used_size - Amount of Space Used on a File System ” on page 160

sto_path_free_sz - Automatic Storage Path Free Space

Element identifier	fs_free_size
Element type	information

Table 50. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	db_sto_path_info	Buffer Pool

Description

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.

Database identification and status monitor elements

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

Related reference:

- “db_storage_path - Automatic storage path ” on page 159
- “fs_id - Unique File System Identification Number ” on page 161
- “fs_used_size - Amount of Space Used on a File System ” on page 160
- “fs_total_size - Total Size of a File System ” on page 160
- “fs_type - File System Type ” on page 162

fs_used_size - Amount of Space Used on a File System

Element identifier fs_used_size

Element type information

Table 51. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	db_sto_path_info	Buffer Pool

Description

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

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

Related reference:

- “db_storage_path - Automatic storage path ” on page 159
- “sto_path_free_sz - Automatic Storage Path Free Space ” on page 159
- “fs_id - Unique File System Identification Number ” on page 161
- “fs_total_size - Total Size of a File System ” on page 160
- “fs_type - File System Type ” on page 162
- “num_db_storage_paths - Number of automatic storage paths ” on page 159

fs_total_size - Total Size of a File System

Element identifier fs_total_size

Element type information

Table 52. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	db_sto_path_info	Buffer Pool

Description

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

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

Related reference:

- “db_storage_path - Automatic storage path ” on page 159
- “sto_path_free_sz - Automatic Storage Path Free Space ” on page 159
- “fs_id - Unique File System Identification Number ” on page 161
- “fs_type - File System Type ” on page 162
- “fs_used_size - Amount of Space Used on a File System ” on page 160

fs_id - Unique File System Identification Number

Element identifier	fs_id
Element type	information

Table 53. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	db_sto_path_info	Buffer Pool

Description

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

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

Related reference:

- “db_storage_path - Automatic storage path ” on page 159
- “sto_path_free_sz - Automatic Storage Path Free Space ” on page 159
- “fs_total_size - Total Size of a File System ” on page 160
- “fs_type - File System Type ” on page 162
- “fs_used_size - Amount of Space Used on a File System ” on page 160

fs_type - File System Type

Element identifier	fs_type
Element type	information

Table 54. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	db_sto_path_info	Buffer Pool

Description

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..

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

Related reference:

- “db_storage_path - Automatic storage path ” on page 159
- “sto_path_free_sz - Automatic Storage Path Free Space ” on page 159
- “fs_id - Unique File System Identification Number ” on page 161
- “fs_total_size - Total Size of a File System ” on page 160
- “fs_used_size - Amount of Space Used on a File System ” on page 160

Application identification and status

Application identification and status monitor elements

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

- agent_id - Application Handle (agent ID) monitor element
- appl_status - Application Status monitor element
- codepage_id - ID of Code Page Used by Application monitor element
- status_change_time - Application Status Change Time monitor element
- appl_id_oldest_xact - Application with Oldest Transaction monitor element
- smallest_log_avail_node - Node with Least Available Log Space monitor element
- appl_name - Application Name monitor element
- appl_id - Application ID monitor element
- sequence_no - Sequence Number monitor element
- auth_id - Authorization ID monitor element
- session_auth_id - Session Authorization ID monitor element
session_auth_id - Session Authorization ID monitor element
- client_nname - Configuration NNAME of Client monitor element
- client_prdid - Client Product/Version ID monitor element

Application identification and status monitor elements

- client_db_alias - Database Alias Used by Application monitor element
- host_prdid - Host Product/Version ID monitor element
- outbound_appl_id - Outbound Application ID monitor element
- outbound_sequence_no - Outbound Sequence Number monitor element
- execution_id - User Login ID monitor element
- corr_token - DRDA Correlation Token monitor element
- client_pid - Client Process ID monitor element
- client_platform - Client Operating Platform monitor element
- client_protocol - Client Communication Protocol monitor element
- territory_code - Database Territory Code monitor element
- appl_priority - Application Agent Priority monitor element
- appl_priority_type - Application Priority Type monitor element
- authority_lvl - User Authorization Level monitor element
- node_number - Node Number monitor element
- data_partition_id - Data Partition Identifier monitor element
data_partition_id - Data Partition Identifier monitor element
- coord_node - Coordinating Node monitor element
- appl_con_time - Connection Request Start Timestamp monitor element
- connections_top - Maximum Number of Concurrent Connections monitor element
- conn_complete_time - Connection Request Completion Timestamp monitor element
- prev_uow_stop_time - Previous Unit of Work Completion Timestamp monitor element
- uow_start_time - Unit of Work Start Timestamp monitor element
- uow_stop_time - Unit of Work Stop Timestamp monitor element
- uow_elapsed_time - Most Recent Unit of Work Elapsed Time monitor element
- uow_comp_status - Unit of Work Completion Status monitor element
- uow_status - Unit of Work Status monitor element
- appl_idle_time - Application Idle Time monitor element
- DB2 agent information monitor elements

agent_id - Application Handle (agent ID)

Element identifier	agent_id
Element type	information

Table 55. 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 56. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

Application identification and status monitor elements

Table 56. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-
Statements	event_subsection	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

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.

Usage The application handle can be used to uniquely identify an active application (application handle is synonymous with agent Id).

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

Element identifier appl_status

Element type information

Table 57. Snapshot Monitoring Information

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

Table 58. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

The current status of the application.

Application identification and status monitor elements

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.
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.

Application identification and status monitor elements

API Constant	Description
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.
SQLM_QUIESCE_TABLESPACE	Quiescing a Table space: The application is performing a quiesce table space request.
SQLM_WAITFOR_REMOTE	Wait for Remote Partition: The application is waiting for a response from a remote partition in a partitioned database instance.
SQLM_REMOTE_RQST	Remote 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.

Related reference:

- "status_change_time - Application Status Change Time " on page 167
- "stmt_operation/operation - Statement Operation " on page 386

codepage_id - ID of Code Page Used by Application

Element identifier	codepage_id
Element type	information

Table 59. 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

Application identification and status monitor elements

Table 60. Event Monitoring Information

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

Description

The code page identifier.

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

Element identifier status_change_time

Element type timestamp

Table 61. 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

Description

The date and time the application entered its current status.

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.

Related reference:

- “appl_status - Application Status ” on page 164

appl_id_oldest_xact - Application with Oldest Transaction

Element identifier appl_id_oldest_xact

Element type information

Table 62. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Application identification and status monitor elements

Description

The application ID (which corresponds to the *agent_id* value from the application snapshot) of the application that has the oldest transaction.

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.

Related reference:

- "agent_id_holding_lock - Agent ID Holding Lock " on page 321
- "deadlocks - Deadlocks Detected " on page 303

smallest_log_avail_node - Node with Least Available Log Space

Element identifier smallest_log_avail_node

Element type information

Table 63. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

This element is only returned for global snapshots and indicates the node with the least amount (in bytes) of available log space.

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.

Related reference:

- "appl_id_oldest_xact - Application with Oldest Transaction " on page 167
- "total_log_used - Total Log Space Used " on page 292
- "total_log_available - Total Log Available " on page 293

appl_name - Application Name

Element identifier appl_name

Element type information

Table 64. 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

Application identification and status monitor elements

Table 65. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

Description

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

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*.

Related reference:

- “*appl_id* - Application ID ” on page 169
- “*codepage_id* - ID of Code Page Used by Application ” on page 166

appl_id - Application ID

Element identifier	appl_id
Element type	information

Table 66. 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 67. 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	-

Description

This identifier is generated when the application connects to the database at the database manager or when DDCS receives a request to connect to a DRDA database.

Application identification and status monitor elements

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 DDCS applications, you will also need to use `outbound_appl_id` 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, DDCS, or both, are running. Each of the formats consists of three parts separated by periods.

1. APPC

Format Network.LU Name.Application instance

Example CAIBMTOR.OSFDBX0.930131194520

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 create 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.

2. 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".

Application identification and status monitor elements

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.

3. IPX/SPX

Format Netid.nodeid.Application instance

Example C11A8E5C.400011528250.0131214645

Details An IPX/SPX-generated application ID is made up by concatenating a character network ID (8 hexadecimal characters), a node id (12 hexadecimal characters), and a unique identifier for the instance of the application. The application instance corresponds to a 10-decimal-character time stamp of the form MDDHHMSS.

4. NetBIOS

Format *NETBIOS.nname.Application instance

Example *NETBIOS.SB0IVIN.930131214645

Details For non-partitioned database systems, a NetBIOS application ID is made up by concatenating the string "*NETBIOS", the nname defined in the client's database configuration file, and a unique identifier for the instance of this application. For partitioned database systems, a NetBIOS application ID is made up by concatenating the string "Nxxx.etc" where xxx is the partition the application is attached to.

5. 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 partition instances, LOCAL is replaced with Nx, where x is the partition number from which the client connected to the database. For example, *N2.DB2INST1.0B5A1222841.

Application identification and status monitor elements

Use *client_protocol* to determine which communications protocol the connection is using and, as a result, the format of the *appl_id*.

Related reference:

- “outbound_appl_id - Outbound Application ID ” on page 175
- “client_protocol - Client Communication Protocol ” on page 178

sequence_no - Sequence Number

Element identifier sequence_no

Element type information

Table 68. Snapshot Monitoring Information

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

Table 69. 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_dlcomm event_stmt_history	-
Deadlocks with Details History Values	event_detailed_dlcomm event_stmt_history	-

Description

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.

auth_id - Authorization ID

Element identifier auth_id

Element type information

Table 70. 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 71. 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.

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

Related reference:

- "appl_name - Application Name " on page 168

session_auth_id - Session Authorization ID

Element identifier session_auth_id

Element type information

Table 72. Snapshot Monitoring Information

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

Description

The current authorization ID for the session being used by this application.

Usage You can use this element to determine what authorization ID is being used to prepare SQL statements, execute SQL statements, or both.

client_nname - Configuration NNAME of Client

Element identifier client_nname

Element type information

Table 73. Snapshot Monitoring Information

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

Table 74. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

Description

The *nname* in the database manager configuration file at the client database partition.

Usage You can use this element to identify the client database partition that is running the application. This element only applies to Windows Environments where the NetBIOS LAN environment exists.

Application identification and status monitor elements

Related reference:

- “server_nname - Configuration NNAME at Monitoring (Server) Database Partition ” on page 149

client_prdid - Client Product/Version ID

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	-

Description

The product and version that is running on the client.

- Usage** You can use this element to identify the product and code version of the database 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-digit modification level.

Related reference:

- “server_prdid - Server Product/Version ID ” on page 150

client_db_alias - Database Alias Used by Application

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	-

Description

The alias of the database provided by the application to connect to the database.

- Usage** This element can be used to identify the actual database that the

Application identification and status monitor elements

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.

Related reference:

- “last_reset - Last Reset Timestamp ” on page 419
- “input_db_alias - Input Database Alias ” on page 420
- “db_name - Database Name ” on page 154

host_prdid - Host Product/Version ID

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

Description

The product and version that is running on the server.

Usage Used to identify the product and code version of the DRDA host database product. It is in the form PPPVVRRM, where:

- PPP identifies the host DRDA product
 - ARI for DB2 Server for VSE & VM
 - DSN for DB2 for OS/390® and z/OS®
 - QSQ for DB2 UDB for AS/400
 - 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-digit modification level

outbound_appl_id - Outbound Application ID

Element identifier	outbound_appl_id
Element type	information

Table 80. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic

Description

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* is used to connect a client to the DB2 Connect gateway.

Application identification and status monitor elements

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.

Related reference:

- “*appl_id* - Application ID ” on page 169

outbound_sequence_no - Outbound Sequence Number

Element identifier outbound_sequence_no

Element type information

Table 81. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic

Description

This element will be blank when the gateway concentrator is on, or if the DCS application is not in a logical unit of work.

execution_id - User Login ID

Element identifier execution_id

Element type information

Table 82. Snapshot Monitoring Information

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

Table 83. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

Description

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.

Usage You can use this element to determine the operating system userid of the individual running the application that you are monitoring.

Related reference:

- “`auth_id` - Authorization ID ” on page 172

corr_token - DRDA Correlation Token

Element identifier `corr_token`
Element type information

Table 84. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	<code>appl_info</code>	Basic
Application	<code>appl</code>	Basic

Table 85. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	<code>event_connheader</code>	-

Description

The DRDA AS correlation token.

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

Element identifier `client_pid`
Element type information

Table 86. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	<code>appl_info</code>	Basic
Application	<code>appl</code>	Basic
DCS Application	<code>dcs_appl_info</code>	Basic

Application identification and status monitor elements

Table 87. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

Description

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

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

Element identifier client_platform

Element type information

Table 88. Snapshot Monitoring Information

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

Table 89. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

Description

The operating system on which the client application is running.

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*.

Related reference:

- "server_platform - Server Operating System " on page 152

client_protocol - Client Communication Protocol

Element identifier client_protocol

Element type information

Table 90. Snapshot Monitoring Information

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

Table 91. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

Description

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

Usage This element can be used for problem determination for remote applications. Values for this field are:

API Constant	Communication Protocol
SQLM_PROT_UNKNOWN	(note 1)
SQLM_PROT_LOCAL	none (note 2)
SQLM_PROT_APPC	APPC
SQLM_PROT_TCPIP	TCP/IP
SQLM_PROT_IPXSPX	IPX/SPX
SQLM_PROT_NETBIOS	NETBIOS

Notes:

1. The client is communicating using an unknown protocol. This value will only be returned if future clients connect with a down-level server.
2. The client is running on the same node as the server and no communications protocol is in use.

territory_code - Database Territory Code

Element identifier territory_code

Element type information

Table 92. Snapshot Monitoring Information

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

Table 93. Event Monitoring Information

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

Description

The territory code of the database for which the monitor data is collected. This monitor element was formerly known as country_code.

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

Element identifier appl_priority

Element type information

Table 94. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Application identification and status monitor elements

Table 95. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

The priority of the agents working for this application.

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.

Related reference:

- "appl_priority_type - Application Priority Type " on page 180

appl_priority_type - Application Priority Type

Element identifier appl_priority_type

Element type information

Table 96. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 97. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

Operating system priority type for the agent working on behalf of the application.

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

Related reference:

- "query_cost_estimate - Query Cost Estimate " on page 396
- "appl_priority - Application Agent Priority " on page 179

authority_lvl - User Authorization Level

Element identifier authority_lvl

Application identification and status monitor elements

Element type information

Table 98. Snapshot Monitoring Information

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

Table 99. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

The highest authority level granted to an application.

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

Related concepts:

- “Authorization, privileges, and object ownership” in *Administration Guide: Implementation*

node_number - Node Number

Element identifier node_number

Element type information

Application identification and status monitor elements

Table 100. Snapshot Monitoring Information

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

Table 101. 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	-

Description

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

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

coord_node - Coordinating Node

Element identifier coord_node

Element type information

Table 102. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 103. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

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

Usage Each connected application is served by one coordinator node.

appl_con_time - Connection Request Start Timestamp

Element identifier	appl_con_time
Element type	timestamp

Table 104. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Timestamp

Description

The date and time that an application started a connection request.

Usage Use this element to determine when the application started its connection request to the database.

Related reference:

- “conn_complete_time - Connection Request Completion Timestamp ” on page 184

connections_top - Maximum Number of Concurrent Connections

Element identifier	connections_top
Element type	water mark

Table 105. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 106. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The highest number of simultaneous connections to the database since the database was activated.

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:

$$\text{rem_cons_in} + \text{local_cons}$$

Related reference:

- “rem_cons_in - Remote Connections To Database Manager ” on page 191
- “local_cons - Local Connections ” on page 192

conn_complete_time - Connection Request Completion Timestamp

Element identifier	conn_complete_time
Element type	timestamp

Table 107. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Timestamp

Description

The date and time that a connection request was granted.

Usage Use this element to determine when a connection request to the database was granted.

Related reference:

- “appl_con_time - Connection Request Start Timestamp ” on page 183

prev_uow_stop_time - Previous Unit of Work Completion Timestamp

Element identifier	prev_uow_stop_time
Element type	timestamp

Table 108. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Unit of Work, Timestamp
DCS Application	dcs_appl	Unit of Work, Timestamp

Description

This is the time the unit of work completed.

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.

Related reference:

- “uow_start_time - Unit of Work Start Timestamp ” on page 185
- “uow_stop_time - Unit of Work Stop Timestamp ” on page 185
- “conn_complete_time - Connection Request Completion Timestamp ” on page 184

uow_start_time - Unit of Work Start Timestamp

Element identifier	uow_start_time
Element type	timestamp

Table 109. 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

Description

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

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.

Related reference:

- "uow_stop_time - Unit of Work Stop Timestamp " on page 185
- "prev_uow_stop_time - Previous Unit of Work Completion Timestamp " on page 184
- "conn_complete_time - Connection Request Completion Timestamp " on page 184

uow_stop_time - Unit of Work Stop Timestamp

Element identifier	uow_stop_time
Element type	timestamp

Table 110. 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

Application identification and status monitor elements

Description

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

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*.

Related reference:

- “*uow_start_time* - Unit of Work Start Timestamp ” on page 185
- “*prev_uow_stop_time* - Previous Unit of Work Completion Timestamp ” on page 184
- “*conn_complete_time* - Connection Request Completion Timestamp ” on page 184

uow_elapsed_time - Most Recent Unit of Work Elapsed Time

Element identifier uow_elapsed_time

Element type time

Table 111. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Unit of Work, Timestamp
DCS Application	dcs_appl	Unit of Work, Timestamp

Description

The elapsed execution time of the most recently completed unit of work.

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

Related reference:

- “*gw_comm_errors* - Communication Errors ” on page 471
- “*gw_comm_error_time* - Communication Error Time ” on page 471

uow_comp_status - Unit of Work Completion Status

Element identifier uow_comp_status

Element type information

Application identification and status monitor elements

Table 112. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Unit of Work
DCS Application	dcs_appl	Basic

Table 113. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Transactions	event_xact	-

Description

The status of the unit of work and how it stopped.

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

Element identifier	uow_status
Element type	information

Table 114. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Transactions	event_xact	-

Description

The status of the unit of work.

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

Element identifier	appl_idle_time
Element type	information

Table 115. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement
DCS Application	dcs_appl	Statement

Application identification and status monitor elements

Description

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.

Usage This information can be used to implement applications that force users that have been idle for a specified number of seconds.

Related reference:

- “query_cost_estimate - Query Cost Estimate ” on page 396
- “appl_priority - Application Agent Priority ” on page 179
- “appl_priority_type - Application Priority Type ” on page 180

data_partition_id - Data Partition Identifier monitor element

Element identifier data_partition_id

Element type information

Table 116. Snapshot monitoring information

Snapshot level	Logical data grouping	Monitor switch
Table	table	Basic
Lock	lock	Lock
Lock	lock_wait	Lock

Table 117. Event monitoring information

Event type	Logical data grouping	Monitor switch
Table	event_table	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-
Deadlocks	lock	-

Description

The identifier of the data partition for which information is returned.

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.

Related tasks:

- “Using snapshot monitor data to monitor the reorganization of a partitioned table” on page 29

DB2 agent information

DB2 agent information monitor elements

The following database system monitor elements provide information about agents:

Application identification and status monitor elements

- agent_pid - Process or Thread ID monitor element
- coord_agent_pid - Coordinator Agent monitor element

agent_pid - Process or Thread ID

Element identifier	agent_pid
Element type	information

Table 118. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	agent	Statement

Description

The process Id (UNIX systems) or thread Id (Windows systems) of a DB2 agent.

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.

Related reference:

- “coord_agent_pid - Coordinator Agent ” on page 189

coord_agent_pid - Coordinator Agent

Element identifier	coord_agent_pid
Element type	information

Table 119. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic

Description

The process Id (UNIX systems) or thread Id (Windows systems) of the coordinator agent for the application.

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

Related reference:

- “agent_pid - Process or Thread ID ” on page 189

Database manager configuration

Database manager configuration monitor elements

The following elements provide database manager configuration information.

- Agents and connections monitor elements
- Memory pool monitor elements
- Sort monitor elements
- Hash join monitor elements
- Fast communications manager monitor elements
- Utilities monitor elements

Agents and connections

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. Creating an agent, if an idle agent is not available.
3. Finding an agent that is associated with another application. For example, if an agent cannot be created because *maxagents* has been reached, DB2 will try to take an idle agent associated with another application. This is referred to as a **stolen agent**.

- **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 *maxagents* configuration parameter defines the maximum number of agents, regardless of type, that can exist for an instance. The *maxagents* value does not create any agents. 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. If subagents are not used, *max_coordagents* equals *maxagents*. If subagents are used, some combination of coordinator agents and subagents could reach *maxagents*.

When an agent is assigned work, it attempts to obtain a token or permission to process the transaction. The database manager controls the number of tokens available using the *maxcagents* configuration parameter. If a token is not available, the agent will sleep until one becomes available, at which time the requested work will be processed. This allows you to use *maxcagents* to control the load, or number of concurrently executing transactions, the server handles.

Database manager identification and status monitor elements

The following elements provide agent and connection information:

- `rem_cons_in` - Remote Connections To Database Manager monitor element
- `rem_cons_in_exec` - Remote Connections Executing in the Database Manager monitor element
- `local_cons` - Local Connections monitor element
- `local_cons_in_exec` - Local Connections Executing in the Database Manager monitor element
- `con_local_dbases` - Local Databases with Current Connects monitor element
- `total_cons` - Connects Since Database Activation monitor element
- `appls_cur_cons` - Applications Connected Currently monitor element
- `appls_in_db2` - Applications Executing in the Database Currently monitor element
- `agents_registered` - Agents Registered monitor element
- `agents_waiting_on_token` - Agents Waiting for a Token monitor element
- `agents_registered_top` - Maximum Number of Agents Registered monitor element
- `agents_waiting_top` - Maximum Number of Agents Waiting monitor element
- `idle_agents` - Number of Idle Agents monitor element
- `agents_from_pool` - Agents Assigned From Pool monitor element
- `agents_created_empty_pool` - Agents Created Due to Empty Agent Pool monitor element
- `coord_agents_top` - Maximum Number of Coordinating Agents monitor element
- `agents_stolen` - Stolen Agents monitor element
- `associated_agents_top` - Maximum Number of Associated Agents monitor element
- `comm_private_mem` - Committed Private Memory monitor element
- `total_sec_cons` - Secondary Connections monitor element
- `num_assoc_agents` - Number of Associated Agents monitor element
- `max_agent_overflows` - Maximum Agent Overflows monitor element
- `num_gw_conn_switches` - Connection Switches monitor element

`rem_cons_in` - Remote Connections To Database Manager

Element identifier `rem_cons_in`

Element type gauge

Table 120. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The current number of connections initiated from remote clients to the instance of the database manager that is being monitored.

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` configuration parameter, described in the *Administration Guide*.

Related reference:

Database manager identification and status monitor elements

- “rem_cons_in_exec - Remote Connections Executing in the Database Manager ” on page 192
- “local_cons - Local Connections ” on page 192
- “local_cons_in_exec - Local Connections Executing in the Database Manager ” on page 193

rem_cons_in_exec - Remote Connections Executing in the Database Manager

Element identifier rem_cons_in_exec

Element type gauge

Table 121. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

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.

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 *maxcagents* configuration parameter, described in the *Administration Guide*.

Related reference:

- “rem_cons_in - Remote Connections To Database Manager ” on page 191
- “local_cons - Local Connections ” on page 192
- “local_cons_in_exec - Local Connections Executing in the Database Manager ” on page 193

local_cons - Local Connections

Element identifier local_cons

Element type gauge

Table 122. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

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

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.

Database manager identification and status monitor elements

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 `maxagents` configuration parameter, described in the *Administration Guide*.

Related reference:

- “`rem_cons_in` - Remote Connections To Database Manager ” on page 191
- “`rem_cons_in_exec` - Remote Connections Executing in the Database Manager ” on page 192
- “`local_cons_in_exec` - Local Connections Executing in the Database Manager ” on page 193

`local_cons_in_exec` - Local Connections Executing in the Database Manager

Element identifier `local_cons_in_exec`
Element type gauge

Table 123. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

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.

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 `maxagents` configuration parameter, described in the *Administration Guide*.

Related reference:

- “`rem_cons_in` - Remote Connections To Database Manager ” on page 191
- “`rem_cons_in_exec` - Remote Connections Executing in the Database Manager ” on page 192
- “`local_cons` - Local Connections ” on page 192

`con_local_dbases` - Local Databases with Current Connects

Element identifier `con_local_dbases`
Element type gauge

Table 124. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Database manager identification and status monitor elements

Description

The number of local databases that have applications connected.

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

Element identifier total_cons

Element type counter

Table 125. 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 126. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

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

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.

Related reference:

- “db_conn_time - Database Activation Timestamp ” on page 155
- “appls_cur_cons - Applications Connected Currently ” on page 194
- “appls_in_db2 - Applications Executing in the Database Currently ” on page 195
- “total_sec_cons - Secondary Connections ” on page 200

appls_cur_cons - Applications Connected Currently

Element identifier appls_cur_cons

Element type gauge

Table 127. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Database manager identification and status monitor elements

Table 127. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	db_lock_list	Basic

Description

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

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, which are described in the *Administration Guide*. 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.

Related reference:

- “*appls_in_db2* - Applications Executing in the Database Currently ” on page 195
- “*total_cons* - Connects Since Database Activation ” on page 194
- “*rem_cons_in* - Remote Connections To Database Manager ” on page 191
- “*local_cons* - Local Connections ” on page 192

appls_in_db2 - Applications Executing in the Database Currently

Element identifier appls_in_db2

Element type gauge

Table 128. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

Indicates the number of applications that are currently connected to the database, and for which the database manager is currently processing a request.

Usage You can use this element to understand how many of the database manager agent tokens are being used by applications connected to this database. If the sum of *rem_cons_in_exec* and *local_cons_in_exec* is equal to the value of the *maxcagents* configuration parameter, you may want to increase the value of that parameter, as described in the *Administration Guide*.

Related reference:

- “*appls_cur_cons* - Applications Connected Currently ” on page 194
- “*total_cons* - Connects Since Database Activation ” on page 194
- “*rem_cons_in_exec* - Remote Connections Executing in the Database Manager ” on page 192
- “*local_cons_in_exec* - Local Connections Executing in the Database Manager ” on page 193
- “*locks_waiting* - Current Agents Waiting On Locks ” on page 320

agents_registered - Agents Registered

Element identifier agents_registered

Element type gauge

Table 129. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

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

Usage You can use this element to help evaluate your setting for the *maxagents* configuration parameter.

Related reference:

- “agents_registered_top - Maximum Number of Agents Registered ” on page 196

agents_waiting_on_token - Agents Waiting for a Token

Element identifier agents_waiting_on_token

Element type gauge

Table 130. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The number of agents waiting for a token so they can execute a transaction in the database manager.

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*. For more information about this parameter, see the *Administration Guide*.

Related reference:

- “agents_registered - Agents Registered ” on page 196

agents_registered_top - Maximum Number of Agents Registered

Element identifier agents_registered_top

Element type water mark

Table 131. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Database manager identification and status monitor elements

Description

The maximum number of agents that the database manager has ever registered, at the same time, since it was started (coordinator agents and subagents).

Usage You may use this element to help you evaluate your setting of the *maxagents* configuration parameter, described in the *Administration Guide*.

The number of agents registered at the time the snapshot was taken is recorded by *agents_registered*.

Related reference:

- “agents_registered - Agents Registered ” on page 196
- “agents_waiting_top - Maximum Number of Agents Waiting ” on page 197

agents_waiting_top - Maximum Number of Agents Waiting

Element identifier agents_waiting_top

Element type water mark

Table 132. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The maximum number of agents that have ever been waiting for a token, at the same time, since the database manager was started.

Usage You may use this element to help you evaluate your setting of the *maxcagents* configuration parameter, described in the *Administration Guide*.

The number of agents waiting for a token at the time the snapshot was taken is recorded by *agents_waiting_on_token*.

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.

Related reference:

- “agents_waiting_on_token - Agents Waiting for a Token ” on page 196
- “agents_registered_top - Maximum Number of Agents Registered ” on page 196

idle_agents - Number of Idle Agents

Element identifier idle_agents

Element type gauge

Table 133. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The number of agents in the agent pool that are currently unassigned to an application and are, therefore, “idle”.

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. See the *Administration Guide* for more information.

Database manager identification and status monitor elements

Related reference:

- “agents_registered_top - Maximum Number of Agents Registered ” on page 196
- “agents_waiting_top - Maximum Number of Agents Waiting ” on page 197
- “agents_registered - Agents Registered ” on page 196

agents_from_pool - Agents Assigned From Pool

Element identifier agents_from_pool

Element type counter

Table 134. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The number of agents assigned from the agent pool.

Usage This element can be used with *agents_created_empty_pool* to determine how often an agent must be created because the pool is empty.

If the ratio of

$\text{Agents Created Due to Empty Agent Pool} / \text{Agents Assigned From Pool}$

is high, it may indicate that the *num_poolagents* configuration parameter should be increased. A low ratio suggests that *num_poolagents* is set too high, and that some of the agents in the pool are rarely used and are wasting system resources.

A high ratio can indicate that the overall workload for this node is too high. You can adjust the workload by lowering the maximum number of coordinating agents specified by the *maxcagents* configuration parameter, or by redistributing data among the nodes.

See the *Administration Guide* for more information on the Agent Pool Size (*num_poolagents*) and Maximum Number of Concurrent Coordinating Agents (*maxcagents*) configuration parameters.

Related reference:

- “agents_created_empty_pool - Agents Created Due to Empty Agent Pool ” on page 198
- “coord_agents_top - Maximum Number of Coordinating Agents ” on page 199

agents_created_empty_pool - Agents Created Due to Empty Agent Pool

Element identifier agents_created_empty_pool

Element type counter

Table 135. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The number of agents created because the agent pool was empty. It includes the number of agents started at DB2 start up (*num_initagents*).

Database manager identification and status monitor elements

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.

Related reference:

- “`agents_from_pool` - Agents Assigned From Pool ” on page 198
- “`coord_agents_top` - Maximum Number of Coordinating Agents ” on page 199

`coord_agents_top` - Maximum Number of Coordinating Agents

Element identifier `coord_agents_top`

Element type water mark

Table 136. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Database	dbase	Basic

Description

The maximum number of coordinating agents working at one time.

Usage If the peak number of coordinating agents represents too high a workload for this node, you can reduce the number that can be concurrently executing a transaction by changing the `maxcagents` configuration parameter.

See the *Administration Guide* for more information on the Maximum Number of Concurrent Coordinating Agents (`maxcagents`) configuration parameter.

Related reference:

- “`agents_from_pool` - Agents Assigned From Pool ” on page 198
- “`agents_created_empty_pool` - Agents Created Due to Empty Agent Pool ” on page 198

`agents_stolen` - Stolen Agents

Element identifier `agents_stolen`

Element type counter

Table 137. 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.

Description

The number of times that agents are stolen from an application. Agents are stolen when an idle agent associated with an application is reassigned to work on a different application.

Usage This element can be used in conjunction with `associated_agents_top` to evaluate the load that this application places on the system.

Database manager identification and status monitor elements

If *agents_stolen* is high, consider increasing the *num_poolagents* configuration parameter.

Related reference:

- “num_agents - Number of Agents Working on a Statement ” on page 411

associated_agents_top - Maximum Number of Associated Agents

Element identifier associated_agents_top

Element type water mark

Table 138. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Description

The maximum number of subagents associated with this application.

Usage If the peak number of subagents is close to the *num_poolagents* configuration parameter, this might indicate too high a workload for this node.

See the *Administration Guide* for more information on the Agent Pool Size (*num_poolagents*) configuration parameter.

Related reference:

- “agents_from_pool - Agents Assigned From Pool ” on page 198
- “agents_created_empty_pool - Agents Created Due to Empty Agent Pool ” on page 198

comm_private_mem - Committed Private Memory

Element identifier comm_private_mem

Element type gauge

Table 139. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The amount of private memory that the instance of the database manager has currently committed at the time of the snapshot.

Usage You can use this element to help set the *min_priv_mem* configuration parameter (see the *Administration Guide*) to ensure you have enough private memory available. This element is returned for all platforms, but tuning can only be accomplished on platforms where DB2 uses threads (such as Windows 2000).

total_sec_cons - Secondary Connections

Element identifier total_sec_cons

Element type counter

Database manager identification and status monitor elements

Table 140. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The number of connections made by a subagent to the database at the node.

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.

Related reference:

- “total_cons - Connects Since Database Activation ” on page 194
- “db2start_time - Start Database Manager Timestamp ” on page 148
- “db_conn_time - Database Activation Timestamp ” on page 155

num_assoc_agents - Number of Associated Agents

Element identifier num_assoc_agents

Element type gauge

Table 141. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl_info	Basic

Description

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.

Usage You can use this element to help evaluate your settings for your agent configuration parameters.

Related reference:

- “agents_from_pool - Agents Assigned From Pool ” on page 198
- “agents_created_empty_pool - Agents Created Due to Empty Agent Pool ” on page 198
- “associated_agents_top - Maximum Number of Associated Agents ” on page 200
- “max_agent_overflows - Maximum Agent Overflows ” on page 201

max_agent_overflows - Maximum Agent Overflows

Element identifier max_agent_overflows

Element type gauge

Table 142. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Database manager identification and status monitor elements

Description

The number of times a request to create a new agent was received when the *maxagents* configuration parameter had already been reached.

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.

See the *Administration Guide* for more information on the Maximum Number of Agents (*maxagents*) configuration parameter.

Related reference:

- “agents_from_pool - Agents Assigned From Pool ” on page 198
- “agents_created_empty_pool - Agents Created Due to Empty Agent Pool ” on page 198
- “associated_agents_top - Maximum Number of Associated Agents ” on page 200
- “num_assoc_agents - Number of Associated Agents ” on page 201

num_gw_conn_switches - Connection Switches

Element identifier num_gw_conn_switches

Element type gauge

Table 143. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The number of times that an agent from the agents pool was primed with a connection and was stolen for use with a different DRDA database.

Usage Use this element to determine if the size of the agent pool should be increased.

Related reference:

- “agents_registered - Agents Registered ” on page 196
- “max_agent_overflows - Maximum Agent Overflows ” on page 201
- “agents_registered_top - Maximum Number of Agents Registered ” on page 196
- “num_assoc_agents - Number of Associated Agents ” on page 201
- “total_sec_cons - Secondary Connections ” on page 200

Memory pool

Memory pool monitor elements

The following elements provide information about the memory pools:

- pool_id - Memory Pool Identifier monitor element
- pool_secondary_id - Memory Pool Secondary Identifier monitor element
- pool_cur_size - Current Size of Memory Pool monitor element
- pool_config_size - Configured Size of Memory Pool monitor element
- pool_watermark - Memory Pool Watermark monitor element

The nature of memory_pool data elements varies between platforms: on Windows systems, the database system monitor does not report any memory in database snapshots, while on UNIX systems, memory is reported in database snapshots.

Database manager identification and status monitor elements

Instead of reporting this memory in database snapshots, the system monitor for Windows systems reports it in database manager snapshots. This difference in reporting is due to differences in the underlying memory architecture between Windows systems and UNIX systems.

pool_id - Memory Pool Identifier

Element identifier	pool_id
Element type	Information

Table 144. Snapshot Monitoring Information

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

Table 145. Event Monitoring Information

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

Description

The type of memory pool.

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 many of each of the following pools can be expected.

API Constant	Description
SQLM_HEAP_APPLICATION	Application Heap
SQLM_HEAP_DATABASE	Database Heap
SQLM_HEAP_APPL_CONTROL	Application Control 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_DFM	DFM Heap
SQLM_HEAP_QUERY	Query 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

Database manager identification and status monitor elements

API Constant	Description
SQLM_HEAP_APP_GROUP	Application Group Shared Heap
SQLM_HEAP_SHARED_SORT	Sort Shared Heap

Related reference:

- “pool_cur_size - Current Size of Memory Pool ” on page 204
- “pool_config_size - Configured Size of Memory Pool ” on page 205
- “pool_watermark - Memory Pool Watermark ” on page 206

pool_secondary_id - Memory Pool Secondary Identifier

Element identifier pool_secondary_id

Element type Information

Table 146. Snapshot Monitoring Information

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

Table 147. Event Monitoring Information

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

Description

An additional identifier to help determine the memory pool for which monitor data is returned.

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

Related reference:

- “pool_id - Memory Pool Identifier ” on page 203

pool_cur_size - Current Size of Memory Pool

Element identifier pool_cur_size

Database manager identification and status monitor elements

Element type Information

Table 148. Snapshot Monitoring Information

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

Table 149. Event Monitoring Information

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

Description

The current size of a memory pool.

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.

Related reference:

- “pool_id - Memory Pool Identifier ” on page 203
- “pool_config_size - Configured Size of Memory Pool ” on page 205
- “pool_watermark - Memory Pool Watermark ” on page 206

pool_config_size - Configured Size of Memory Pool

Element identifier pool_config_size

Element type Information

Table 150. Snapshot Monitoring Information

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

Table 151. Event Monitoring Information

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

Description

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

Database manager identification and status monitor elements

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.

Related reference:

- “*pool_id* - Memory Pool Identifier ” on page 203
- “*pool_cur_size* - Current Size of Memory Pool ” on page 204
- “*pool_watermark* - Memory Pool Watermark ” on page 206

pool_watermark - Memory Pool Watermark

Element identifier pool_watermark

Element type Information

Table 152. Snapshot Monitoring Information

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

Table 153. Event Monitoring Information

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

Description

The largest size of a memory pool since its creation.

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.

Related reference:

- “*pool_id* - Memory Pool Identifier ” on page 203
- “*pool_cur_size* - Current Size of Memory Pool ” on page 204
- “*pool_config_size* - Configured Size of Memory Pool ” on page 205

Sort

Sort monitor elements

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

- `sort_heap_allocated` - Total Sort Heap Allocated monitor element
- `post_threshold_sorts` - Post Threshold Sorts monitor element
- `pipedsorts_requested` - Piped Sorts Requested monitor element
- `pipedsorts_accepted` - Piped Sorts Accepted monitor element
- `total_sorts` - Total Sorts monitor element
- `total_sort_time` - Total Sort Time monitor element
- `sort_overflows` - Sort Overflows monitor element
- `active_sorts` - Active Sorts monitor element
- `sort_shrheap_allocated` - Sort Share Heap Currently Allocated monitor element
- `sort_shrheap_top` - Sort Share Heap High Water Mark monitor element

`sort_heap_allocated` - Total Sort Heap Allocated

Element identifier `sort_heap_allocated`

Element type gauge

Table 154. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
Database	dbase	Basic

Description

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.

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.

Database manager identification and status monitor elements

Related reference:

- “total_sorts - Total Sorts ” on page 210

post_threshold_sorts - Post Threshold Sorts

Element identifier post_threshold_sorts

Element type counter

Table 155. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Sort

For snapshot monitoring, this counter can be reset.

Description

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

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 (*sheaphthes* 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 (*sheaphthes*) or,
- Adjust applications to use fewer or smaller sorts via SQL query changes.

Related reference:

- “stmt_sorts - Statement Sorts ” on page 394
- “active_sorts - Active Sorts ” on page 212

pipeds_sorts_requested - Piped Sorts Requested

Element identifier pipeds_sorts_requested

Element type counter

Table 156. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

For snapshot monitoring, this counter can be reset.

Description

The number of piped sorts that have been requested.

Usage Each active sort on the system allocates memory, which may result in sorting taking up too much of the available system memory.

Database manager identification and status monitor elements

The sort list heap (*sortheap*) and sort heap threshold (*sheapthres*) configuration parameters help to control the amount of memory used for sort operations. These parameters are also used to determine whether a sort will be piped.

Since piped sorts may reduce disk I/O, allowing more piped sorts can improve the performance of sort operations and possibly the performance of the overall system. A piped sort is not accepted if the sort heap threshold will be exceeded when the sort heap is allocated for the sort. See *pipedsorts_accepted* for more information if you are experiencing piped sort rejections.

The SQL EXPLAIN output will show whether the optimizer requests a piped sort. For more information on piped and non-piped sorts see the *Administration Guide*.

Related reference:

- “pipedsorts_accepted - Piped Sorts Accepted ” on page 209
- “post_threshold_sorts - Post Threshold Sorts ” on page 208

pipedsorts_accepted - Piped Sorts Accepted

Element identifier	pipedsorts_accepted
Element type	counter

Table 157. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

For snapshot monitoring, this counter can be reset.

Description

The number of piped sorts that have been accepted.

Usage Each active sort on the system allocates memory, which may result in sorting taking up too much of the available system memory.

When the number of accepted piped sorts is low compared to the number requested, you can improve sort performance by adjusting one or both of the following configuration parameters:

- sortheap
- sheapthres

If piped sorts are being rejected, you might consider decreasing your sort heap or increasing your sort heap threshold. You should be aware of the possible implications of either of these options. If you increase the sort heap threshold, then there is the possibility that more memory will remain allocated for sorting. This could cause the paging of memory to disk. If you decrease the sort heap, you might require an extra merge phase that could slow down the sort.

See the *Administration Guide* for more information on sorts.

Related reference:

- “pipedsorts_requested - Piped Sorts Requested ” on page 208
- “post_threshold_sorts - Post Threshold Sorts ” on page 208

Database manager identification and status monitor elements

total_sorts - Total Sorts

Element identifier total_sorts

Element type counter

Table 158. 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 159. Event Monitoring Information

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

Description

The total number of sorts that have been executed.

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.

Related reference:

- “sort_overflows - Sort Overflows ” on page 211
- “total_sort_time - Total Sort Time ” on page 210

total_sort_time - Total Sort Time

Element identifier total_sort_time

Element type counter

Table 160. 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.

Database manager identification and status monitor elements

Table 161. Event Monitoring Information

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

Description

The total elapsed time (in milliseconds) for all sorts that have been executed.

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.
```

Related reference:

- “total_sorts - Total Sorts ” on page 210
- “sort_overflows - Sort Overflows ” on page 211

sort_overflows - Sort Overflows

Element identifier sort_overflows

Element type counter

Table 162. 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.

Database manager identification and status monitor elements

Table 163. Event Monitoring Information

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

Description

The total number of sorts that ran out of sort heap and may have required disk space for temporary storage.

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.

Related reference:

- “total_sorts - Total Sorts ” on page 210

active_sorts - Active Sorts

Element identifier active_sorts

Element type gauge

Table 164. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The number of sorts in the database that currently have a sort heap allocated.

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.

Related reference:

- “sort_heap_allocated - Total Sort Heap Allocated ” on page 207
- “total_sorts - Total Sorts ” on page 210

sort_heap_top - Sort Private Heap High Water Mark

Element identifier sort_heap_top

Element type water mark

Table 165. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

The private sort memory high-water mark, in 4k pages, across the database manager.

Usage This element can be used to determine if the SHEAPTHRES configuration parameter is set to an optimal value. For example, if this water mark 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

Element identifier sort_shrheap_allocated

Element type information

Table 166. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

Total amount of shared sort memory allocated in the database.

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 Water Mark

Element identifier sort_shrheap_top

Element type water mark

Table 167. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

Database-wide shared sort memory high-water mark in 4k pages.

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-water mark is persistently much lower than the shared sort memory

Database manager identification and status monitor elements

threshold, it is likely that this threshold needs to be decreased, thus freeing memory for other database functions. Conversely, if this high-water mark begins to approach the shared sort memory threshold, then this might indicate that this threshold needs to be increased. This is important because the shared sort memory threshold is a hard limit. When the total amount of sort memory reaches this threshold, no more shared sorts can be initiated.

This element, along with the high-water mark 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-water marks, 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-water mark.

post_shrthreshold_sorts - Post threshold sorts

Element identifier	post_shrthreshold_sorts
Element type	counter

Table 168. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Sort

For snapshot monitoring, this counter can be reset.

Table 169. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

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.

Related reference:

- “active_sorts - Active Sorts ” on page 212
- “sheapthres - Sort heap threshold configuration parameter” in *Performance Guide*
- “sheapthres_shr - Sort heap threshold for shared sorts configuration parameter” in *Performance Guide*

Hash join

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 monitor element
- post_threshold_hash_joins - Hash Join Threshold monitor element
- total_hash_loops - Total Hash Loops monitor element
- hash_join_overflows - Hash Join Overflows monitor element
- hash_join_small_overflows - Hash Join Small Overflows monitor element

active_hash_joins - Active hash joins

Element identifier active_hash_joins

Element type counter

Table 170. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	-

Table 171. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The total number of hash joins that are currently running and consuming memory.

Related reference:

- “post_shrthreshold_hash_joins - Post threshold hash joins ” on page 216

total_hash_joins - Total Hash Joins

Element identifier total_hash_joins

Element type counter

Table 172. 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 173. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Database manager identification and status monitor elements

Table 173. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

The total number of hash joins executed.

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.

Related reference:

- “post_threshold_hash_joins - Hash Join Threshold ” on page 216
- “total_hash_loops - Total Hash Loops ” on page 217
- “hash_join_overflows - Hash Join Overflows ” on page 218
- “hash_join_small_overflows - Hash Join Small Overflows ” on page 218

post_threshold_hash_joins - Hash Join Threshold

Element identifier post_threshold_hash_joins

Element type counter

Table 174. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

For snapshot monitoring, this counter can be reset.

Description

The total number of times that a hash join heap request was limited due to concurrent use of shared or private sort heap space.

Usage If this value is large (greater than 5% of hash_join_overflows), the sort heap threshold should be increased.

Related reference:

- “total_hash_joins - Total Hash Joins ” on page 215
- “total_hash_loops - Total Hash Loops ” on page 217
- “hash_join_overflows - Hash Join Overflows ” on page 218
- “hash_join_small_overflows - Hash Join Small Overflows ” on page 218

post_shrthreshold_hash_joins - Post threshold hash joins

Element identifier post_shrthreshold_hash_joins

Element type counter

Table 175. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	-

For snapshot monitoring, this counter can be reset.

Table 176. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

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. 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.

Related reference:

- “active_hash_joins - Active hash joins ” on page 215
- “sheapthres - Sort heap threshold configuration parameter” in *Performance Guide*
- “sheapthres_shr - Sort heap threshold for shared sorts configuration parameter” in *Performance Guide*

total_hash_loops - Total Hash Loops

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	-

Description

The total number of times that a single partition of a hash join was larger than the available sort heap space.

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.

Related reference:

- “total_hash_joins - Total Hash Joins ” on page 215
- “post_threshold_hash_joins - Hash Join Threshold ” on page 216
- “hash_join_overflows - Hash Join Overflows ” on page 218

Database manager identification and status monitor elements

- “hash_join_small_overflows - Hash Join Small Overflows ” on page 218

hash_join_overflows - Hash Join Overflows

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	-

Description

The number of times that hash join data exceeded the available sort heap space.

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.

Related reference:

- “total_hash_joins - Total Hash Joins ” on page 215
- “post_threshold_hash_joins - Hash Join Threshold ” on page 216
- “total_hash_loops - Total Hash Loops ” on page 217
- “hash_join_small_overflows - Hash Join Small Overflows ” on page 218

hash_join_small_overflows - Hash Join Small Overflows

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	-

Database manager identification and status monitor elements

Description

The number of times that hash join data exceeded the available sort heap space by less than 10%.

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.

Related reference:

- “total_hash_joins - Total Hash Joins ” on page 215
- “post_threshold_hash_joins - Hash Join Threshold ” on page 216
- “total_hash_loops - Total Hash Loops ” on page 217
- “hash_join_overflows - Hash Join Overflows ” on page 218

Fast communications manager

Fast communications manager monitor elements

The following database system monitor elements provide information about the Fast Communication Manager (FCM):

- `buff_free` - FCM Buffers Currently Free monitor element
- `buff_free_bottom` - Minimum FCM Buffers Free monitor element
- `ch_free` - Channels Currently Free monitor element
- `ch_free_bottom` - Minimum Channels Free monitor element
- `connection_status` - Connection Status monitor element
- `total_buffers_sent` - Total FCM Buffers Sent monitor element
- `total_buffers_rcvd` - Total FCM Buffers Received monitor element

`buff_free` - FCM Buffers Currently Free

Element identifier `buff_free`

Element type `gauge`

Table 183. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description

This element indicates the number of FCM buffers currently free.

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`.

Related reference:

- “buff_free_bottom - Minimum FCM Buffers Free ” on page 219

`buff_free_bottom` - Minimum FCM Buffers Free

Element identifier `buff_free_bottom`

Element type `water mark`

Database manager identification and status monitor elements

Table 184. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description

The lowest number of free FCM buffers reached during processing.

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.

Related reference:

- “*buff_free* - FCM Buffers Currently Free ” on page 219

ch_free - Channels Currently Free

Element identifier ch_free

Element type gauge

Table 185. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description

This element indicates the number of inter-node communication channels that are currently free.

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*.

Related reference:

- “*ch_free_bottom* - Minimum Channels Free ” on page 220
- “*fcm_num_channels* - Number of FCM channels configuration parameter” in *Performance Guide*

ch_free_bottom - Minimum Channels Free

Element identifier ch_free_bottom

Element type water mark

Table 186. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description

The lowest number of free inter-node communication channels reached during processing.

Usage Use this element in conjunction with the *fcm_num_channels* configuration parameter to determine the maximum connection entry utilization.

Database manager identification and status monitor elements

Related reference:

- “ch_free - Channels Currently Free ” on page 220
- “fcm_num_channels - Number of FCM channels configuration parameter” in *Performance Guide*

connection_status - Connection Status

Element identifier connection_status

Element type information

Table 187. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm_node	Basic

Description

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.

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.

Related reference:

- “total_buffers_sent - Total FCM Buffers Sent ” on page 221
- “total_buffers_rcvd - Total FCM Buffers Received ” on page 222

total_buffers_sent - Total FCM Buffers Sent

Element identifier total_buffers_sent

Element type counter

Table 188. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm_node	Basic

Description

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).

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.

Related reference:

Database manager identification and status monitor elements

- “connection_status - Connection Status ” on page 221
- “total_buffers_rcvd - Total FCM Buffers Received ” on page 222

total_buffers_rcvd - Total FCM Buffers Received

Element identifier	total_buffers_rcvd
Element type	counter

Table 189. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm_node	Basic

Description

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).

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.

Related reference:

- “connection_status - Connection Status ” on page 221
- “total_buffers_sent - Total FCM Buffers Sent ” on page 221

Utilities

Utilities monitor elements

The following elements provide information about utilities:

- utility_dbname - Database Operated on by Utility monitor element
utility_dbname - Database Operated on by Utility monitor element
- utility_id - Utility ID monitor element
utility_id - Utility ID monitor element
- utility_type - Utility Type monitor element
utility_type - Utility Type monitor element
- utility_priority - Utility Priority monitor element
utility_priority - Utility Priority monitor element
- utility_start_time - Utility Start Time monitor element
utility_start_time - Utility Start Time monitor element
- utility_description - Utility Description monitor element
utility_description - Utility Description monitor element
- progress_list_cur_seq_num - Current Progress List Sequence Number monitor element
progress_list_cur_seq_num - Current Progress List Sequence Number monitor element
- progress_list_attr - Current Progress List Attributes monitor element
progress_list_attr - Current Progress List Attributes monitor element
- progress_seq_num - Progress Sequence Number monitor element
progress_seq_num - Progress Sequence Number monitor element
- progress_description - Progress Description monitor element
progress_description - Progress Description monitor element
- progress_start_time - Progress Start Time monitor element
progress_start_time - Progress Start Time monitor element
- progress_work_metric - Progress Work Metric monitor element
progress_work_metric - Progress Work Metric monitor element

Database manager identification and status monitor elements

- progress_total_units - Total Progress Work Units monitor element
elementprogress_total_units - Total Progress Work Units monitor element
- progress_completed_units - Completed Progress Work Units monitor element
elementprogress_completed_units - Completed Progress Work Units monitor element

utility_dbname - Database Operated on by Utility

Element identifier utility_dbname

Element type information

Table 190. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

The database operated on by the utility.

utility_id - Utility ID

Element identifier utility_id

Element type information

Table 191. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

The unique identifier corresponding to the utility invocation.

utility_type - Utility Type

Element identifier utility_type

Element type information

Table 192. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

The class of utility.

Usage

The values for this element, listed as follows, are defined in sqlmon.h.

API Constant	Utility
SQLM_UTILITY_REBALANCE	Rebalance
SQLM_UTILITY_BACKUP	Backup
SQLM_UTILITY_RUNSTATS	Runstats
SQLM_UTILITY_REORG	Reorg
SQLM_UTILITY_RESTORE	Restore
SQLM_UTILITY_CRASH_RECOVERY	Crash recovery

Database manager identification and status monitor elements

API Constant	Utility
SQLM_UTILITY_ROLLFORWARD_RECOVERY	Rollforward recovery
SQLM_UTILITY_LOAD	Load
SQLM_UTILITY_RESTART_RECREATE_INDEX	Restart recreate index

utility_priority - Utility Priority

Element identifier	utility_priority
Element type	information

Table 193. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

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.

Related reference:

- “LIST UTILITIES command” in *Command Reference*
- “SET UTIL_IMPACT_PRIORITY command” in *Command Reference*
- “util_impact_lim - Instance impact policy configuration parameter” in *Performance Guide*

utility_start_time - Utility Start Time

Element identifier	utility_start_time
Element type	timestamp

Table 194. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

The date and time when the current utility was originally invoked.

utility_description - Utility Description

Element identifier	utility_description
Element type	information

Table 195. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

A brief description of the work a utility is performing. For example, a rebalance invocation may contain “Tablespace ID: 2” representing that this

Database manager identification and status monitor elements

rebalancer is working on tablespace with ID 2. The format of this field is dependent on the class of utility and is subject to change between releases.

progress_list_cur_seq_num - Current Progress List Sequence Number

Element identifier progress_list_cur_seq_num

Element type information

Table 196. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress_list	Basic

Description

If the utility contains multiple sequential phases, then this element displays the number of the current phase.

Usage Use this element to determine the current phase of a multiphase utility. See description of *progress_seq_num*.

Related reference:

- “progress_seq_num - Progress Sequence Number ” on page 225

progress_list_attr - Current Progress List Attributes

Element identifier progress_list_attr

Element type information

Table 197. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress list	Basic

Description

This element describes how to interpret a a list of progress elements. 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.

Usage Use this element to determine how the elements of a progress_list will be updated.

Related reference:

- “progress_list_cur_seq_num - Current Progress List Sequence Number ” on page 225

progress_seq_num - Progress Sequence Number

Element identifier progress_seq_num

Element type information

Database manager identification and status monitor elements

Table 198. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress	Basic

Description

Phase number.

Note: The phase number displays only for utilities that consist of multiple phases of execution.

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*.

Related reference:

- “*progress_list_cur_seq_num* - Current Progress List Sequence Number ” on page 225

progress_description - Progress Description

Element identifier progress_description

Element type information

Table 199. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress	Basic

Description

Describes the phase of work. 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

Element identifier progress_start_time

Element type information

Table 200. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress	Basic

Description

A timestamp representing the start of the phase.

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

Element identifier progress_work_metric

Database manager identification and status monitor elements

Element type information

Table 201. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress	Basic

Description

The metric for interpreting the *progress_total_units* and *progress_completed_units* elements. Example values include:

- SQLM_WORK_METRIC_BYTES
- SQLM_WORK_METRIC_EXTENTS

Notes:

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.

Related reference:

- “*progress_completed_units* - Completed Progress Work Units ” on page 227
- “*progress_total_units* - Total Progress Work Units ” on page 227

progress_total_units - Total Progress Work Units

Element identifier *progress_total_units*

Element type information

Table 202. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress	Basic

Description

Total amount of work to perform in order for the phase to be complete. 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:

$$\text{percentage complete} = \text{progress_completed_units} / \text{progress_total_units} * 100$$

Related reference:

- “*progress_completed_units* - Completed Progress Work Units ” on page 227
- “*progress_work_metric* - Progress Work Metric ” on page 226

progress_completed_units - Completed Progress Work Units

Element identifier *progress_completed_units*

Element type information

Database manager identification and status monitor elements

Table 203. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	progress	Basic

Description

The number of work units for the current phase which have been completed. 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).

Notes:

1. This element might not be included for all utilities.
2. This element is expressed in units displayed by the *progress_work_metric* monitor element.

Usage Use this element to determine the amount of completed work within a phase. By itself, this element can be used to monitor the activity of a running utility. This element should constantly increase as the utility executes. If the *progress_completed_units* fails to increase over a long period of time then the utility might be stalled.

If *progress_total_units* is defined, then this element can be used to calculate the percentage of completed work:

percentage complete = $\text{progress_completed_units} / \text{progress_total_units} * 100$

Related reference:

- “*progress_total_units* - Total Progress Work Units ” on page 227
- “*progress_work_metric* - Progress Work Metric ” on page 226

utility_state - Utility State

Element identifier utility_state

Element type information

Table 204. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

This element describes the state of a utility.

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

Related reference:

- “*utility_dbname* - Database Operated on by Utility ” on page 223
- “*utility_description* - Utility Description ” on page 224

Database manager identification and status monitor elements

- “utility_id - Utility ID ” on page 223
- “utility_type - Utility Type ” on page 223

utility_invoker_type - Utility Invoker Type

Element identifier utility_invoker_type
Element type information

Table 205. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	utility_info	Basic

Description

This element describes how a utility was invoked.

Usage Use this element to determine how a utility was invoked. For example, you can use it to determine whether a utility was invoked automatically by DB2 or by a user. The values for this element, listed as follows, are defined in sqlmon.h.

API Constant	Utility
SQLM_UTILITY_INVOKER_USER	Utility was invoked by user
SQLM_UTILITY_INVOKER_AUTO	Utility was invoked automatically by DB2

Related reference:

- “utility_dbname - Database Operated on by Utility ” on page 223
- “utility_id - Utility ID ” on page 223

Database configuration

Database configuration monitor elements

The following elements provide information particularly helpful for database performance tuning.

- Buffer pool activity monitor elements
- Non-buffered I/O activity monitor elements
- Catalog cache monitor elements
- Package cache monitor elements
- SQL workspaces monitor elements
- Database heap monitor elements
- Logging monitor elements

Buffer pool activity

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:

Database configuration monitor elements

- 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 ID monitor element
- pool_data_l_reads - Buffer Pool Data Logical Reads monitor element
- pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads monitor element
- pool_data_p_reads - Buffer Pool Data Physical Reads monitor element
- pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads monitor element
- pool_data_writes - Buffer Pool Data Writes monitor element
- pool_index_l_reads - Buffer Pool Index Logical Reads monitor element
- pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads monitor element
- pool_index_p_reads - Buffer Pool Index Physical Reads monitor element
- pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads monitor element
- pool_index_writes - Buffer Pool Index Writes monitor element
- pool_xda_l_reads - Buffer Pool XDA Data Logical Reads monitor element
- pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads monitor element
- pool_xda_p_reads - Buffer Pool XDA Data Physical Reads monitor element
- pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element
- pool_xda_writes - Buffer Pool XDA Data Writes monitor element
- pool_read_time - Total Buffer Pool Physical Read Time monitor element

Database configuration monitor elements

- pool_write_time - Total Buffer Pool Physical Write Time monitor element
- files_closed - Database Files Closed monitor element
- pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element
- pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element
- pool_async_index_writes - Buffer Pool Asynchronous Index Writes monitor element
- pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element
- pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes monitor element
- pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads monitor element
- pool_async_read_time - Buffer Pool Asynchronous Read Time monitor element
- pool_async_write_time - Buffer Pool Asynchronous Write Time monitor element
- pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests monitor element
- pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests monitor element
- pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests monitor element
- pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered monitor element
- pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered monitor element
- pool_no_victim_buffer - Buffer Pool No Victim Buffers monitor element
- pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered monitor element
- bp_name - Buffer Pool Name monitor element
- prefetch_wait_time - Time Waited for Prefetch monitor element
- unread_prefetch_pages - Unread Prefetch Pages monitor element
- pages_from_vectored_ios - Total Number of Pages Read by Vectored IO monitor element
- block_ios - Number of Block IO Requests monitor element
- vectored_ios - Number of Vectored IO Requests monitor element
- pages_from_block_ios - Total Number of Pages Read by Block IO monitor element
- physical_page_maps - Number of Physical Page Maps monitor element

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

bp_id - Buffer Pool ID

Element identifier	bp_id
Element type	information

Table 206. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Basic

Description

This element contains the buffer pool identifier for the buffer pool that is being monitored.

pool_data_l_reads - Buffer Pool Data Logical Reads

Element identifier pool_data_l_reads

Element type counter

Table 207. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool
Application	appl	Buffer Pool
Application	stmt	Buffer Pool
Dynamic SQL	dynsql	Buffer Pool, Statement

For snapshot monitoring, this counter can be reset.

Table 208. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespace	-
Connection	event_conn	-
Statement	event_stmt	-

Description

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.

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:

$$1 - (\text{pool_data_p_reads} / \text{pool_data_l_reads})$$

For information about determining the overall buffer pool hit ratio see *Buffer pool activity monitor elements*.

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

Database configuration monitor elements

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.

Related reference:

- “pool_data_p_reads - Buffer Pool Data Physical Reads ” on page 235
- “pool_data_writes - Buffer Pool Data Writes ” on page 237
- “pool_index_l_reads - Buffer Pool Index Logical Reads ” on page 238
- “pool_index_p_reads - Buffer Pool Index Physical Reads ” on page 240
- “pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads ” on page 234
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “db_conn_time - Database Activation Timestamp ” on page 155
- “Buffer pool activity monitor elements” on page 229

pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads

Element identifier pool_temp_data_l_reads

Element type counter

Table 209. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool
Application	appl	Buffer Pool
Application	stmt	Buffer Pool
Dynamic SQL	dynsql	Buffer Pool, Statement

For snapshot monitoring, this counter can be reset.

Table 210. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespace	-
Connection	event_conn	-
Statement	event_stmt	-

Description

Indicates the number of data pages which have been requested from the buffer pool (logical) for temporary table spaces.

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage

In conjunction with the *pool_temp_data_p_reads* element, a calculation for the data page hit ratio for buffer pools located in temporary tablespaces 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*.

Related reference:

- “pool_data_l_reads - Buffer Pool Data Logical Reads ” on page 233
- “pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads ” on page 236
- “pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads ” on page 239
- “pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads ” on page 241
- “Buffer pool activity monitor elements” on page 229

pool_data_p_reads - Buffer Pool Data Physical Reads

Element identifier pool_data_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	-

Description

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.

Usage See *pool_data_l_reads* and *pool_async_data_reads* for information about how to use this element.

Related reference:

Database configuration monitor elements

- “pool_async_data_reads - Buffer Pool Asynchronous Data Reads ” on page 245
- “pool_data_l_reads - Buffer Pool Data Logical Reads ” on page 233
- “pool_data_writes - Buffer Pool Data Writes ” on page 237
- “pool_index_l_reads - Buffer Pool Index Logical Reads ” on page 238
- “pool_index_p_reads - Buffer Pool Index Physical Reads ” on page 240
- “pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads ” on page 236
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “db_conn_time - Database Activation Timestamp ” on page 155

pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads

Element identifier pool_temp_data_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_tablespace	-
Connection	event_conn	-
Statement	event_stmt	-

Description

Indicates the number of data pages read in from the table space containers (physical) for temporary table spaces.

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage See *pool_temp_data_l_reads* for information about how to use this element.

Related reference:

- “pool_data_p_reads - Buffer Pool Data Physical Reads ” on page 235
- “pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads ” on page 234
- “pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads ” on page 239
- “pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads ” on page 241

pool_data_writes - Buffer Pool Data Writes

Element identifier pool_data_writes

Element type counter

Table 215. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool
Application	appl	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 216. Event Monitoring Information

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

Description

Indicates the number of times a buffer pool data page was physically written to disk.

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.

Database configuration monitor elements

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.

Related reference:

- “db_conn_time - Database Activation Timestamp ” on page 155
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “pool_data_l_reads - Buffer Pool Data Logical Reads ” on page 233
- “pool_data_p_reads - Buffer Pool Data Physical Reads ” on page 235
- “pool_write_time - Total Buffer Pool Physical Write Time ” on page 243
- “pool_async_data_writes - Buffer Pool Asynchronous Data Writes ” on page 246

pool_index_l_reads - Buffer Pool Index Logical Reads

Element identifier pool_index_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	-

Description

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.

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:

$$1 - (\text{pool_index_p_reads} / \text{pool_index_l_reads})$$

To calculate the overall buffer pool hit ratio, see `pool_data_l_reads`.

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.

Related reference:

- “`pool_data_l_reads` - Buffer Pool Data Logical Reads ” on page 233
- “`pool_data_p_reads` - Buffer Pool Data Physical Reads ” on page 235
- “`pool_data_writes` - Buffer Pool Data Writes ” on page 237
- “`pool_index_p_reads` - Buffer Pool Index Physical Reads ” on page 240
- “`pool_index_writes` - Buffer Pool Index Writes ” on page 241
- “`pool_temp_index_p_reads` - Buffer Pool Temporary Index Physical Reads ” on page 241
- “`appl_con_time` - Connection Request Start Timestamp ” on page 183
- “`db_conn_time` - Database Activation Timestamp ” on page 155

pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads

Element identifier `pool_temp_index_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	-

Description

Indicates the number of index pages which have been requested from the buffer pool (logical) for temporary table spaces.

Database configuration monitor elements

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage See *pool_temp_data_l_reads* for information about how to use this element.

Related reference:

- “pool_index_p_reads - Buffer Pool Index Physical Reads ” on page 240
- “pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads ” on page 234
- “pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads ” on page 236
- “pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads ” on page 241

pool_index_p_reads - Buffer Pool Index Physical Reads

Element identifier pool_index_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_tablespaces	-
Connection	event_conn	-
Statement	event_stmt	-

Description

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.

Usage See *pool_index_l_reads* for information about how to use this element.

Related reference:

- “pool_data_l_reads - Buffer Pool Data Logical Reads ” on page 233
- “pool_data_p_reads - Buffer Pool Data Physical Reads ” on page 235
- “pool_index_l_reads - Buffer Pool Index Logical Reads ” on page 238
- “pool_index_writes - Buffer Pool Index Writes ” on page 241

- “pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads ” on page 239
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “db_conn_time - Database Activation Timestamp ” on page 155

pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads

Element identifier pool_temp_index_p_reads
 Element type counter

Table 223. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool
Application	appl	Buffer Pool
Application	stmt	Buffer Pool
Dynamic SQL	dynsql	Buffer Pool, Statement

For snapshot monitoring, this counter can be reset.

Table 224. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespaces	-
Connection	event_conn	-
Statement	event_stmt	-

Description

Indicates the number of index pages read in from the table space containers (physical) for temporary table spaces.

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage See *pool_temp_data_l_reads* for information about how to use this element.

Related reference:

- “pool_index_l_reads - Buffer Pool Index Logical Reads ” on page 238
- “pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads ” on page 234
- “pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads ” on page 236
- “pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads ” on page 239

pool_index_writes - Buffer Pool Index Writes

Element identifier pool_index_writes
 Element type counter

Database configuration monitor elements

Table 225. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool
Application	appl	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 226. Event Monitoring Information

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

Description

Indicates the number of times a buffer pool index page was physically written to disk.

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.

Related reference:

- “db_conn_time - Database Activation Timestamp ” on page 155
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “pool_index_l_reads - Buffer Pool Index Logical Reads ” on page 238
- “pool_index_p_reads - Buffer Pool Index Physical Reads ” on page 240
- “pool_async_index_writes - Buffer Pool Asynchronous Index Writes ” on page 247

pool_read_time - Total Buffer Pool Physical Read Time

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	-

Description

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.

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*.

Related reference:

- “pool_data_p_reads - Buffer Pool Data Physical Reads ” on page 235
- “pool_index_p_reads - Buffer Pool Index Physical Reads ” on page 240
- “db_conn_time - Database Activation Timestamp ” on page 155
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “pool_async_read_time - Buffer Pool Asynchronous Read Time ” on page 249

pool_write_time - Total Buffer Pool Physical Write Time

Element identifier pool_write_time

Element type counter

Database configuration monitor elements

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	-

Description

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.

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*.

Related reference:

- “pool_data_writes - Buffer Pool Data Writes ” on page 237
- “pool_index_writes - Buffer Pool Index Writes ” on page 241
- “db_conn_time - Database Activation Timestamp ” on page 155
- “appl_con_time - Connection Request Start Timestamp ” on page 183

files_closed - 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	-

Description

The total number of database files closed.

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

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	-

Description

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.

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:

$$\text{pool_data_p_reads} + \text{pool_temp_data_p_reads} - \text{pool_async_data_reads}$$

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*.

Related reference:

- “pool_async_read_time - Buffer Pool Asynchronous Read Time ” on page 249
- “pool_data_p_reads - Buffer Pool Data Physical Reads ” on page 235

Database configuration monitor elements

- “pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests ” on page 250
- “direct_reads - Direct Reads From Database ” on page 269

pool_async_data_writes - Buffer Pool Asynchronous Data Writes

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	-

Description

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.

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*.

Related reference:

- “pool_async_index_writes - Buffer Pool Asynchronous Index Writes ” on page 247
- “pool_data_writes - Buffer Pool Data Writes ” on page 237
- “pool_async_write_time - Buffer Pool Asynchronous Write Time ” on page 249
- “pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered ” on page 251
- “pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered ” on page 252
- “pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered ” on page 254
- “direct_writes - Direct Writes to Database ” on page 269

pool_async_index_writes - Buffer Pool Asynchronous Index Writes

Element identifier	pool_async_index_writes
Element type	counter

Table 237. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 238. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespace	-

Description

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.

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*.

Related reference:

- “pool_async_data_writes - Buffer Pool Asynchronous Data Writes ” on page 246
- “pool_async_index_reads - Buffer Pool Asynchronous Index Reads ” on page 248
- “pool_index_writes - Buffer Pool Index Writes ” on page 241
- “pool_async_write_time - Buffer Pool Asynchronous Write Time ” on page 249
- “pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered ” on page 251
- “pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered ” on page 252
- “pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered ” on page 254
- “direct_writes - Direct Writes to Database ” on page 269

pool_async_index_reads - Buffer Pool Asynchronous Index Reads

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	-

Description

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.

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:

$$\text{pool_index_p_reads} + \text{pool_temp_index_p_reads} - \text{pool_async_index_reads}$$

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

Asynchronous reads are performed by database manager prefetchers. For information about these prefetchers, see the *Administration Guide*.

Related reference:

- “pool_async_data_writes - Buffer Pool Asynchronous Data Writes ” on page 246
- “pool_async_index_writes - Buffer Pool Asynchronous Index Writes ” on page 247
- “pool_index_p_reads - Buffer Pool Index Physical Reads ” on page 240
- “pool_async_read_time - Buffer Pool Asynchronous Read Time ” on page 249
- “pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered ” on page 251
- “pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered ” on page 252
- “pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered ” on page 254
- “direct_reads - Direct Reads From Database ” on page 269

pool_async_read_time - Buffer Pool Asynchronous Read Time

Element identifier pool_async_read_time

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	-

Description

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.

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.

Related reference:

- “pool_async_data_reads - Buffer Pool Asynchronous Data Reads ” on page 245
- “pool_read_time - Total Buffer Pool Physical Read Time ” on page 243
- “pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests ” on page 250
- “direct_read_time - Direct Read Time ” on page 272

pool_async_write_time - Buffer Pool Asynchronous Write Time

Element identifier pool_async_write_time

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

Database configuration monitor elements

For snapshot monitoring, this counter can be reset.

Table 244. Event Monitoring Information

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

Description

The total elapsed time spent writing data or index pages from the buffer pool to disk by database manager page cleaners.

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.

Related reference:

- “pool_async_data_writes - Buffer Pool Asynchronous Data Writes ” on page 246
- “pool_async_index_writes - Buffer Pool Asynchronous Index Writes ” on page 247
- “pool_write_time - Total Buffer Pool Physical Write Time ” on page 243
- “pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests ” on page 250
- “direct_write_time - Direct Write Time ” on page 272

pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests

Element identifier pool_async_data_read_reqs

Element type counter

Table 245. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 246. Event Monitoring Information

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

Description

The number of asynchronous read requests.

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

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

This average can help you determine the amount of asynchronous I/O done in each interaction with the prefetcher.

Related reference:

- “pool_async_data_reads - Buffer Pool Asynchronous Data Reads ” on page 245

pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests

Element identifier pool_async_index_read_reqs

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	-

Description

The number of asynchronous read requests for index pages.

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.

Related reference:

- “pool_async_index_reads - Buffer Pool Asynchronous Index Reads ” on page 248

pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered

Element identifier pool_lsn_gap_clns

Element type counter

Table 249. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool

For snapshot monitoring, this counter can be reset.

Database configuration monitor elements

Table 250. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The number of times a page cleaner was invoked because the logging space used had reached a predefined criterion for the database.

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.

Related reference:

- “pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered ” on page 252
- “pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered ” on page 254
- “Performance variables” in *Performance Guide*
- “chngpgs_thresh - Changed pages threshold configuration parameter” in *Performance Guide*

pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered

Element identifier pool_drty_pg_steal_clns

Element type counter

Table 251. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	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	-

Description

The number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the database.

Usage Using the following formula, you may calculate what percentage of all cleaner invocations are represented by this element:

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

If this ratio is low, it may indicate that you have defined too many page cleaners. If your *chnngpgs_thresh* is set too low, you may be writing out pages that you will dirty later. Aggressive cleaning defeats one purpose of the buffer pool, that is to defer writing to the last possible moment.

If this ratio is high, it may indicate that you have too few page cleaners defined. Too few page cleaners will increase recovery time after failures (see the *Administration Guide*).

When the DB2_USE_ALTERNATE_PAGE_CLEANING registry variable is OFF:

- The *pool_drty_pg_steal_clns* monitor element is inserted into the monitor stream.
- The *pool_drty_pg_steal_clns* monitor element counts the number of times a page cleaner was invoked because a synchronous write was needed during the victim buffer replacement for the database.

When the DB2_USE_ALTERNATE_PAGE_CLEANING registry variable is ON:

- The *pool_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.

Related reference:

- “chnngpgs_thresh - Changed pages threshold configuration parameter” in *Performance Guide*
- “Performance variables” in *Performance Guide*
- “pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered ” on page 254
- “pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered ” on page 251

Database configuration monitor elements

- “pool_no_victim_buffer - Buffer Pool No Victim Buffers ” on page 254

pool_no_victim_buffer - Buffer Pool No Victim Buffers

Element identifier pool_no_victim_buffer

Element type counter

Table 253. 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 254. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespace	event_tablespace	-

Description

Number of times an agent did not have a preselected victim buffer available.

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.

Related reference:

- “pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered ” on page 252
- “pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered ” on page 254
- “pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered ” on page 251

pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered

Element identifier pool_drty_pg_thrsh_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	-

Description

The number of times a page cleaner was invoked because a buffer pool had reached the dirty page threshold criterion for the database.

Usage The threshold is set by the *chnpggs_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.

Related reference:

- “pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered ” on page 251
- “chnpggs_thresh - Changed pages threshold configuration parameter” in *Performance Guide*
- “Performance variables” in *Performance Guide*

bp_name - Buffer Pool Name

Element identifier bp_name

Element type information

Table 257. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Basic

Database configuration monitor elements

Description

The name of the buffer pool.

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

Element identifier prefetch_wait_time

Element type counter

Table 258. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Application	appl	Buffer Pool

Table 259. Event Monitoring Information

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

Description

The time an application spent waiting for an I/O server (prefetcher) to finish loading pages into the buffer pool.

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

Element identifier unread_prefetch_pages

Element type counter

Table 260. 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 261. Event Monitoring Information

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

Description

Indicates the number of pages that the prefetcher read in that were never used.

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

Element identifier vectored_ios

Element type gauge

Table 262. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

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.

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.

Related reference:

- “block_ios - Number of Block IO Requests ” on page 258
- “pages_from_block_ios - Total Number of Pages Read by Block IO ” on page 259
- “pages_from_vectored_ios - Total Number of Pages Read by Vectored IO ” on page 257

pages_from_vectored_ios - Total Number of Pages Read by Vectored IO

Element identifier pages_from_vectored_ios

Element type gauge

Table 263. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

The total number of pages read by vectored I/O into the page area of the buffer pool.

Related reference:

Database configuration monitor elements

- “block_ios - Number of Block IO Requests ” on page 258
- “vectored_ios - Number of Vectored IO Requests ” on page 257
- “pages_from_block_ios - Total Number of Pages Read by Block IO ” on page 259

block_ios - Number of Block IO Requests

Element identifier block_ios

Element type counter

Table 264. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

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.

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.

Related reference:

- “vectored_ios - Number of Vectored IO Requests ” on page 257
- “pages_from_block_ios - Total Number of Pages Read by Block IO ” on page 259

- “pages_from_vectored_ios - Total Number of Pages Read by Vectored IO ” on page 257

pages_from_block_ios - Total Number of Pages Read by Block IO

Element identifier pages_from_block_ios
Element type counter

Table 265. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

The total number of pages read by block I/O into the block area of the 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.

Related reference:

- “block_ios - Number of Block IO Requests ” on page 258
- “vectored_ios - Number of Vectored IO Requests ” on page 257
- “pages_from_vectored_ios - Total Number of Pages Read by Vectored IO ” on page 257

physical_page_maps - Number of Physical Page Maps

Element identifier physical_page_maps
Element type gauge

Table 266. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

The number of physical page maps.

pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests

Element identifier pool_async_xda_read_reqs
Element type counter

Table 267. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool

Database configuration monitor elements

Table 267. 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 268. Event Monitoring Information

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

Description

The number of asynchronous read requests for XML storage object (XDA) data.

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.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests ” on page 250
- “pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads ” on page 260

pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads

Element identifier pool_async_xda_reads

Element type counter

Table 269. 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 270. Event Monitoring Information

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

Description

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.

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*.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “pool_async_data_reads - Buffer Pool Asynchronous Data Reads ” on page 245
- “pool_xda_p_reads - Buffer Pool XDA Data Physical Reads ” on page 263

pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes

Element identifier pool_async_xda_writes

Element type counter

Table 271. 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 272. Event Monitoring Information

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

Description

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.

Usage

You can use this element with `pool_xda_writes` to calculate the number of physical write requests that were performed synchronously on XML

Database configuration monitor elements

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*.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “pool_async_data_writes - Buffer Pool Asynchronous Data Writes ” on page 246
- “pool_xda_writes - Buffer Pool XDA Data Writes ” on page 264

pool_xda_l_reads - Buffer Pool XDA Data Logical Reads

Element identifier pool_xda_l_reads

Element type counter

Table 273. 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 274. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespace	-
Connection	event_conn	-
Statement	event_stmt	-

Description

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.

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage This count includes accesses to data that is:

- Already in the buffer pool when the database manager needs to process the page
- Read into the buffer pool before the database manager can process the page.

You can use the *pool_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*.

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.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “pool_data_l_reads - Buffer Pool Data Logical Reads ” on page 233
- “pool_xda_p_reads - Buffer Pool XDA Data Physical Reads ” on page 263

pool_xda_p_reads - Buffer Pool XDA Data Physical Reads

Element identifier pool_xda_p_reads

Element type counter

Table 275. 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 276. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Database configuration monitor elements

Table 276. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
Tablespaces	event_tablespace	-
Connection	event_conn	-
Statement	event_stmt	-

Description

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.

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage See `pool_xda_l_reads` and `pool_async_xda_reads` for information about how to use this element.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “`pool_async_xda_reads` - Buffer Pool Asynchronous XDA Data Reads ” on page 260
- “`pool_data_p_reads` - Buffer Pool Data Physical Reads ” on page 235
- “`pool_xda_l_reads` - Buffer Pool XDA Data Logical Reads ” on page 262

pool_xda_writes - Buffer Pool XDA Data Writes

Element identifier `pool_xda_writes`

Element type `counter`

Table 277. 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 278. Event Monitoring Information

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

Description

Indicates the number of times a buffer pool data page for an XML storage object (XDA) was physically written to disk.

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.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “`pool_data_writes` - Buffer Pool Data Writes ” on page 237
- “`pool_xda_l_reads` - Buffer Pool XDA Data Logical Reads ” on page 262
- “`pool_xda_p_reads` - Buffer Pool XDA Data Physical Reads ” on page 263

pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads

Element identifier	pool_temp_xda_l_reads
Element type	counter

Table 279. 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 280. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespace	-
Connection	event_conn	-
Statement	event_stmt	-

Description

Indicates the number of pages for XML storage object (XDA) data which have been requested from the buffer pool (logical) for temporary table spaces.

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage

Database configuration monitor elements

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 tablespaces 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*.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads ” on page 234
- “pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads ” on page 266
- “pool_xda_l_reads - Buffer Pool XDA Data Logical Reads ” on page 262

pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads

Element identifier pool_temp_xda_p_reads

Element type counter

Table 281. 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 282. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tablespaces	event_tablespace	-
Connection	event_conn	-
Statement	event_stmt	-

Description

Indicates the number of pages for XML storage object (XDA) data read in from the table space containers (physical) for temporary table spaces.

The functionality to record buffer pool information at the statement level is supported for API and CLP snapshot requests.

Usage See *pool_temp_xda_l_reads* for information about how to use this element.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “Buffer pool activity monitor elements” on page 229
- “pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads ” on page 236
- “pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads ” on page 265

xda_object_pages - XDA Object Pages

Element identifier xda_object_pages

Element type information

Table 283. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Basic

Table 284. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The number of disk pages consumed by XML storage object (XDA) data.

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.

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related reference:

- “data_object_pages - Data Object Pages ” on page 361

Dynamic buffer pool

bp_cur_buffsz - Current Size of Buffer Pool :

Element identifier bp_cur_buffsz

Element type gauge

Table 285. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool_nodeinfo	Buffer Pool

Description

Current buffer pool size.

Database configuration monitor elements

bp_new_buffsz - New Buffer Pool Size :

Element identifier bp_new_buffsz

Element type information

Table 286. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool_nodeinfo	Buffer Pool

Description

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.

bp_pages_left_to_remove - Number of Pages Left to Remove :

Element identifier bp_pages_left_to_remove

Element type gauge

Table 287. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool_nodeinfo	Buffer Pool

Description

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.

bp_tbsp_use_count - Number of Table Spaces Mapped to Buffer Pool :

Element identifier bp_tbsp_use_count

Element type gauge

Table 288. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool_nodeinfo	Buffer Pool

Description

The number of table spaces using this buffer pool.

Non-buffered I/O activity

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 monitor element
- direct_writes - Direct Writes to Database monitor element
- direct_read_reqs - Direct Read Requests monitor element
- direct_write_reqs - Direct Write Requests monitor element
- direct_read_time - Direct Read Time monitor element
- direct_write_time - Direct Write Time monitor element

direct_reads - Direct Reads From Database

Element identifier direct_reads

Element type counter

Table 289. 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 290. Event Monitoring Information

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

Description

The number of read operations that do not use the buffer pool.

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

Related reference:

- “direct_read_reqs - Direct Read Requests ” on page 270
- “direct_read_time - Direct Read Time ” on page 272
- “direct_writes - Direct Writes to Database ” on page 269

direct_writes - Direct Writes to Database

Element identifier direct_writes

Element type counter

Table 291. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Table Space	tablespace	Buffer Pool
Buffer Pool	bufferpool	Buffer Pool

Database configuration monitor elements

Table 291. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 292. Event Monitoring Information

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

Description

The number of write operations that do not use the buffer pool.

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.

Related reference:

- “direct_write_reqs - Direct Write Requests ” on page 271
- “direct_write_time - Direct Write Time ” on page 272
- “direct_reads - Direct Reads From Database ” on page 269

direct_read_reqs - Direct Read Requests

Element identifier direct_read_reqs

Element type counter

Table 293. 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 294. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Table 294. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-
Tablespaces	event_tablespace	-

Description

The number of requests to perform a direct read of one or more sectors of data.

Usage Use the following formula to calculate the average number of sectors that are read by a direct read:

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

Related reference:

- “direct_reads - Direct Reads From Database ” on page 269
- “direct_read_time - Direct Read Time ” on page 272
- “direct_write_reqs - Direct Write Requests ” on page 271

direct_write_reqs - Direct Write Requests

Element identifier direct_write_reqs

Element type counter

Table 295. 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 296. Event Monitoring Information

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

Description

The number of requests to perform a direct write of one or more sectors of data.

Usage Use the following formula to calculate the average number of sectors that are written by a direct write:

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

Related reference:

- “direct_writes - Direct Writes to Database ” on page 269
- “direct_write_time - Direct Write Time ” on page 272
- “direct_read_reqs - Direct Read Requests ” on page 270

direct_read_time - Direct Read Time

Element identifier direct_read_time

Element type counter

Table 297. 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 298. Event Monitoring Information

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

Description

The elapsed time (in milliseconds) required to perform the direct reads.

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.

Related reference:

- “direct_reads - Direct Reads From Database ” on page 269
- “direct_read_reqs - Direct Read Requests ” on page 270
- “direct_write_time - Direct Write Time ” on page 272

direct_write_time - Direct Write Time

Element identifier direct_write_time

Element type counter

Table 299. 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 300. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Table 300. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-
Tablespaces	event_tablespace	-

Description

The elapsed time (in milliseconds) required to perform the direct writes.

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.

Related reference:

- “direct_writes - Direct Writes to Database ” on page 269
- “direct_write_reqs - Direct Write Requests ” on page 271
- “direct_read_time - Direct Read Time ” on page 272

Catalog cache

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 monitor element
- cat_cache_inserts - Catalog Cache Inserts monitor element
- cat_cache_overflows - Catalog Cache Overflows monitor element
- cat_cache_size_top - Catalog Cache High Water Mark monitor element

cat_cache_lookups - Catalog Cache Lookups

Element identifier	cat_cache_lookups
Element type	counter

Database configuration monitor elements

Table 301. 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 302. Event Monitoring Information

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

Description

The number of times that the catalog cache was referenced to obtain table descriptor information or authorization information.

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.

Related reference:

- “cat_cache_inserts - Catalog Cache Inserts ” on page 274
- “cat_cache_overflows - Catalog Cache Overflows ” on page 275
- “cat_cache_size_top - Catalog Cache High Water Mark ” on page 276
- “ddl_sql_stmts - Data Definition Language (DDL) SQL Statements ” on page 378

cat_cache_inserts - Catalog Cache Inserts

Element identifier cat_cache_inserts

Element type counter

Table 303. 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 304. Event Monitoring Information

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

Description

The number of times that the system tried to insert table descriptor or authorization information into the catalog cache.

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.

Related reference:

- "cat_cache_lookups - Catalog Cache Lookups " on page 273
- "cat_cache_overflows - Catalog Cache Overflows " on page 275
- "cat_cache_size_top - Catalog Cache High Water Mark " on page 276

cat_cache_overflows - Catalog Cache Overflows

Element identifier cat_cache_overflows

Element type counter

Table 305. 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 306. Event Monitoring Information

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

Description

The number of times that the catalog cache overflowed the bounds of its allocated memory.

Usage Use this element with `cat_cache_size_top` to determine whether the size of the catalog cache needs to be increased to avoid overflowing.

Database configuration monitor elements

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 *cat_cache_overflows* 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.

Related reference:

- “cat_cache_lookups - Catalog Cache Lookups ” on page 273
- “cat_cache_inserts - Catalog Cache Inserts ” on page 274
- “cat_cache_size_top - Catalog Cache High Water Mark ” on page 276

cat_cache_size_top - Catalog Cache High Water Mark

Element identifier cat_cache_size_top

Element type watermark

Table 307. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 308. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The largest size reached by the catalog cache.

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 Catalog Cache Overflows to determine if such a condition occurred.

You can determine the minimum size of the catalog cache required by your workload by:

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.

Related reference:

- “cat_cache_overflows - Catalog Cache Overflows ” on page 275

Package cache

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 monitor element
- pkg_cache_inserts - Package Cache Inserts monitor element
- pkg_cache_num_overflows - Package Cache Overflows monitor element
- pkg_cache_size_top - Package Cache High Water Mark monitor element
- appl_section_lookups - Section Lookups monitor element
- appl_section_inserts - Section Inserts monitor element

pkg_cache_lookups - Package Cache Lookups

Element identifier pkg_cache_lookups

Element type counter

Table 309. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

For snapshot monitoring, this counter can be reset.

Table 310. Event Monitoring Information

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

Description

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.

Note: 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.

Usage To calculate the package cache hit ratio use the following formula:

$$1 - (\text{Package Cache Inserts} / \text{Package Cache Lookups})$$

Database configuration monitor elements

The package cache hit ratio tells you whether or not the package cache is being used effectively. If the hit ratio is high (more than 0.8), the cache is performing well. A smaller 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.

Related reference:

- “*pkg_cache_inserts* - Package Cache Inserts ” on page 279
- “*static_sql_stmts* - Static SQL Statements Attempted ” on page 373
- “*dynamic_sql_stmts* - Dynamic SQL Statements Attempted ” on page 373
- “*ddl_sql_stmts* - Data Definition Language (DDL) SQL Statements ” on page 378

pkg_cache_inserts - Package Cache Inserts

Element identifier pkg_cache_inserts

Element type counter

Table 311. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

For snapshot monitoring, this counter can be reset.

Table 312. Event Monitoring Information

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

Description

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.

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.

Related reference:

- "pkg_cache_lookups - Package Cache Lookups" on page 277

pkg_cache_num_overflows - Package Cache Overflows

Element identifier pkg_cache_num_overflows

Element type counter

Table 313. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 314. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The number of times that the package cache overflowed the bounds of its allocated memory.

Usage Use this element with pkg_cache_size_top to determine whether the size of the package cache needs to be increased to avoid overflowing.

Related reference:

Database configuration monitor elements

- “pkg_cache_inserts - Package Cache Inserts ” on page 279
- “ddl_sql_stmts - Data Definition Language (DDL) SQL Statements ” on page 378
- “dynamic_sql_stmts - Dynamic SQL Statements Attempted ” on page 373
- “static_sql_stmts - Static SQL Statements Attempted ” on page 373

pkg_cache_size_top - Package Cache High Water Mark

Element identifier pkg_cache_size_top

Element type water mark

Table 315. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 316. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The largest size reached by the package cache.

Usage This element indicates the maximum number of bytes the package cache required for the workload run against the database since it was activated.

If the package cache overflowed, then this element contains the largest size reached by the package cache during the overflow. Check Package Cache Overflows to determine if such a condition occurred.

You can determine the minimum size of the package cache required by your workload by:

$$\text{maximum package cache size} / 4096$$

Rounding the result up to a whole number, indicates the minimum number of 4K pages required by the package cache to avoid overflow.

Related reference:

- “pkg_cache_num_overflows - Package Cache Overflows ” on page 279

appl_section_lookups - Section Lookups

Element identifier appl_section_lookups

Element type counter

Table 317. 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 318. Event Monitoring Information

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

Description

Lookups of SQL sections by an application from its SQL work area.

Usage Each agent has access to a unique SQL work area where the working copy of any executable section is kept. In partitioned databases, this work area is shared by all non-SMP agents. In other environments and with SMP agents, each agent has its own unique SQL work area.

This counter indicates how many times the SQL work area was accessed by agents for an application. It is a cumulative total of all lookups on all SQL work heaps for agents working for this application.

You can use this element in conjunction with *appl_section_inserts* to tune the size of the heap used for the SQL work area. In partitioned databases this size is controlled by the *app_ctl_heap_sz* configuration parameter. SQL work area size in other database environments uses the *applheapsz* configuration parameter. The size of the SQL work area for SMP agents is controlled by *applheapsz* in all environments.

Related reference:

- “app_ctl_heap_sz - Application control heap size configuration parameter” in *Performance Guide*
- “appl_section_inserts - Section Inserts monitor element” on page 281
- “applheapsz - Application heap size configuration parameter” in *Performance Guide*
- “pkg_cache_inserts - Package Cache Inserts ” on page 279
- “pkg_cache_lookups - Package Cache Lookups ” on page 277

appl_section_inserts - Section Inserts monitor element

Element identifier appl_section_inserts

Element type counter

Table 319. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

Table 320. Event Monitoring Information

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

Description

Inserts of SQL sections by an application from its SQL work area.

Usage The working copy of any executable section is stored in a unique SQL work area. This is a count of when a copy was not available and had to be inserted.

See *appl_section_lookups* for more information on using sections.

Related reference:

Database configuration monitor elements

- “appl_section_lookups - Section Lookups ” on page 280
- “pkg_cache_inserts - Package Cache Inserts ” on page 279
- “pkg_cache_lookups - Package Cache Lookups ” on page 277

SQL workspaces

SQL workspaces monitor elements

When sections are required by an application for the execution of dynamic or static SQL statements, they are placed in the shared workspace or the private workspace as required. The shared workspace exists at the application level and is shared by one or more applications. The private workspace exists at the agent level and there is one private workspace associated with each agent.

Since the shared workspace is shared among many applications, applications with similar environments can share the benefits of another agent’s work. Realized benefits include setup and initialization costs.

The following database system monitor elements are used for SQL workspaces:

- shr_workspace_size_top - Maximum Shared Workspace Size monitor element
- shr_workspace_num_overflows - Shared Workspace Overflows monitor element
- shr_workspace_section_lookups - Shared Workspace Section Lookups monitor element
- shr_workspace_section_inserts - Shared Workspace Section Inserts monitor element
- priv_workspace_size_top - Maximum Private Workspace Size monitor element
- priv_workspace_num_overflows - Private Workspace Overflows monitor element
- priv_workspace_section_lookups - Private Workspace Section Lookups monitor element
- priv_workspace_section_inserts - Private Workspace Section Inserts monitor element

shr_workspace_size_top - Maximum Shared Workspace Size

Element identifier shr_workspace_size_top

Element type water mark

Table 321. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

Table 322. Event Monitoring Information

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

Description

The largest size reached by shared workspaces.

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`.

Related reference:

- “shr_workspace_num_overflows - Shared Workspace Overflows ” on page 283

shr_workspace_num_overflows - Shared Workspace Overflows

Element identifier shr_workspace_num_overflows

Element type counter

Table 323. 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 324. Event Monitoring Information

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

Description

The number of times that shared workspaces overflowed the bounds of their allocated memory.

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.

Related reference:

- “shr_workspace_size_top - Maximum Shared Workspace Size ” on page 282

shr_workspace_section_lookups - Shared Workspace Section Lookups

Element identifier shr_workspace_section_lookups

Element type counter

Database configuration monitor elements

Table 325. 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 326. Event Monitoring Information

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

Description

Lookups of SQL sections by applications in shared workspaces.

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.

Related reference:

- “shr_workspace_section_inserts - Shared Workspace Section Inserts ” on page 284

shr_workspace_section_inserts - Shared Workspace Section Inserts

Element identifier shr_workspace_section_inserts

Element type counter

Table 327. 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 328. Event Monitoring Information

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

Description

Number of inserts of SQL sections by applications into shared workspaces.

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.

Related reference:

- “shr_workspace_section_lookups - Shared Workspace Section Lookups ” on page 283

priv_workspace_size_top - Maximum Private Workspace Size

Element identifier priv_workspace_size_top

Element type water mark

Table 329. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

Table 330. Event Monitoring Information

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

Description

The largest size reached by the Private Workspace.

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.

Related reference:

- “priv_workspace_num_overflows - Private Workspace Overflows ” on page 285

priv_workspace_num_overflows - Private Workspace Overflows

Element identifier priv_workspace_num_overflows

Element type counter

Database configuration monitor elements

Table 331. 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 332. Event Monitoring Information

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

Description

The number of times that the private workspaces overflowed the bounds of its allocated memory.

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.

Related reference:

- “`priv_workspace_size_top` - Maximum Private Workspace Size ” on page 285

priv_workspace_section_lookups - Private Workspace Section Lookups

Element identifier `priv_workspace_section_lookups`

Element type `counter`

Table 333. 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 334. Event Monitoring Information

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

Description

Lookups of SQL sections by an application in its agents' private workspace.

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.

Related reference:

- “`priv_workspace_section_inserts` - Private Workspace Section Inserts ” on page 287

priv_workspace_section_inserts - Private Workspace Section Inserts

Element identifier `priv_workspace_section_inserts`
Element type counter

Table 335. 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 336. Event Monitoring Information

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

Description

Inserts of SQL sections by an application into the private workspace.

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.

Related reference:

- “`priv_workspace_section_lookups` - Private Workspace Section Lookups ” on page 286

Database heap

Database heap monitor elements

The following database system monitor elements are used for database heaps:

- db_heap_top - Maximum Database Heap Allocated monitor element

db_heap_top - Maximum Database Heap Allocated

Element identifier db_heap_top

Element type water mark

Table 337. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 338. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

This element is being maintained for DB2 version compatibility. It now measures memory usage, but not exclusively usage by the database heap.

Logging

Logging monitor elements

The following database system monitor elements are used for logging:

- sec_log_used_top - Maximum Secondary Log Space Used monitor element
- tot_log_used_top - Maximum Total Log Space Used monitor element
- sec_logs_allocated - Secondary Logs Allocated Currently monitor element
- log_reads - Number of Log Pages Read monitor element
- log_writes - Number of Log Pages Written monitor element
- uow_log_space_used - Unit of Work Log Space Used monitor element
- total_log_used - Total Log Space Used monitor element
- total_log_available - Total Log Available monitor element
- log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages monitor element
log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages monitor element
- log_to_redo_for_recovery - Amount of Log to be Redone for Recovery monitor element
log_to_redo_for_recovery - Amount of Log to be Redone for Recovery monitor element
- log_write_time - Log Write Time monitor element
log_write_time - Log Write Time monitor element
- log_read_time - Log Read Time monitor element
log_read_time - Log Read Time monitor element
- num_log_write_io - Number of Log Writes monitor element
num_log_write_io - Number of Log Writes monitor element
- num_log_read_io - Number of Log Reads monitor element
num_log_read_io - Number of Log Reads monitor element

- num_log_part_page_io - Number of Partial Log Page Writes monitor element
elementnum_log_part_page_io - Number of Partial Log Page Writes monitor element
- num_log_buffer_full - Number of Full Log Buffers monitor element
elementnum_log_buffer_full - Number of Full Log Buffers monitor element
- num_log_data_found_in_buffer - Number of Log Data Found In Buffer monitor element
elementnum_log_data_found_in_buffer - Number of Log Data Found In Buffer monitor element
- first_active_log - First Active Log File Number monitor element
first_active_log - First Active Log File Number monitor element
- last_active_log - Last Active Log File Number monitor element
last_active_log - Last Active Log File Number monitor element
- current_active_log - Current Active Log File Number monitor element
elementcurrent_active_log - Current Active Log File Number monitor element
- current_archive_log - Current Archive Log File Number monitor element
elementcurrent_archive_log - Current Archive Log File Number monitor element

sec_log_used_top - Maximum Secondary Log Space Used

Element identifier sec_log_used_top

Element type water mark

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	-

Description

The maximum amount of secondary log space used (in bytes).

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:

- logfilesiz
- 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.

Related reference:

Database configuration monitor elements

- “tot_log_used_top - Maximum Total Log Space Used ” on page 290
- “uow_log_space_used - Unit of Work Log Space Used ” on page 292
- “sec_logs_allocated - Secondary Logs Allocated Currently ” on page 290

tot_log_used_top - Maximum Total Log Space Used

Element identifier tot_log_used_top

Element type water mark

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	-

Description

The maximum amount of total log space used (in bytes).

Usage You can use this element to help evaluate the amount of primary log space that you have allocated. Comparing the value of this element with the amount of primary log space you have allocated can help you to evaluate your configuration parameter settings. Your primary log space allocation can be calculated using the following formula:

$$\text{logprimary} \times \text{logfilsiz} \times 4096 \text{ (see note below)}$$

You can use this element in conjunction with *sec_log_used_top* and *sec_logs_allocated* to show your current dependency on secondary logs.

This value includes space used in both primary and secondary log files.

You may need to adjust the following configuration parameters:

- logfilsiz
- logprimary
- logsecond

For more information, see the *Administration Guide*.

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Related reference:

- “uow_log_space_used - Unit of Work Log Space Used ” on page 292
- “sec_logs_allocated - Secondary Logs Allocated Currently ” on page 290
- “sec_log_used_top - Maximum Secondary Log Space Used ” on page 289

sec_logs_allocated - Secondary Logs Allocated Currently

Element identifier sec_logs_allocated

Element type gauge

Table 343. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The total number of secondary log files that are currently being used for the database.

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*.

Related reference:

- “uow_log_space_used - Unit of Work Log Space Used ” on page 292
- “sec_log_used_top - Maximum Secondary Log Space Used ” on page 289
- “tot_log_used_top - Maximum Total Log Space Used ” on page 290

log_reads - Number of Log Pages Read

Element identifier log_reads

Element type counter

Table 344. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 345. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The number of log pages read from disk by the logger.

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.

Related reference:

- “log_writes - Number of Log Pages Written ” on page 291

log_writes - Number of Log Pages Written

Element identifier log_writes

Element type counter

Database configuration monitor elements

Table 346. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 347. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The number of log pages written to disk by the logger.

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.

Related reference:

- “log_reads - Number of Log Pages Read ” on page 291

uow_log_space_used - Unit of Work Log Space Used

Element identifier uow_log_space_used

Element type gauge

Table 348. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Unit of Work

Table 349. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Transactions	event_xact	-

Description

The amount of log space (in bytes) used in the current unit of work of the monitored application.

Usage You may use this element to understand the logging requirements at the unit of work level.

Related reference:

- “sec_logs_allocated - Secondary Logs Allocated Currently ” on page 290
- “sec_log_used_top - Maximum Secondary Log Space Used ” on page 289
- “tot_log_used_top - Maximum Total Log Space Used ” on page 290

total_log_used - Total Log Space Used

Element identifier total_log_used

Element type gauge

Table 350. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The total amount of active log space currently used (in bytes) in the database.

Usage Use this element in conjunction with `total_log_available` to determine whether you may need to adjust the following configuration parameters to avoid running out of log space:

- `logfilesiz`
- `logprimary`
- `logsecond`

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.

Related reference:

- “`uow_log_space_used` - Unit of Work Log Space Used ” on page 292
- “`sec_logs_allocated` - Secondary Logs Allocated Currently ” on page 290
- “`tot_log_used_top` - Maximum Total Log Space Used ” on page 290
- “`appl_id_oldest_xact` - Application with Oldest Transaction ” on page 167

total_log_available - Total Log Available

Element identifier `total_log_available`

Element type gauge

Table 351. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The amount of active log space in the database that is not being used by uncommitted transactions (in bytes).

Usage Use this element in conjunction with `total_log_used` to determine whether you may need to adjust the following configuration parameters to avoid running out of log space:

- `logfilesiz`
- `logprimary`
- `logsecond`

If `total_log_available` goes down to 0, `SQL0964N` will be returned. You may need to increase the 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`.

Database configuration monitor elements

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Related reference:

- “uow_log_space_used - Unit of Work Log Space Used ” on page 292
- “sec_logs_allocated - Secondary Logs Allocated Currently ” on page 290
- “total_log_used - Total Log Space Used ” on page 292
- “appl_id_oldest_xact - Application with Oldest Transaction ” on page 167

log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages

Element identifier log_held_by_dirty_pages

Element type watermark

Table 352. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 353. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

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.

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.

Related reference:

- “log_to_redo_for_recovery - Amount of Log to be Redone for Recovery ” on page 295
- “logfilsiz - Size of log files configuration parameter” in *Performance Guide*
- “num_iocleaners - Number of asynchronous page cleaners configuration parameter” in *Performance Guide*
- “pool_drty_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered ” on page 252

- “pool_drty_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered ” on page 254
- “pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered ” on page 251
- “softmax - Recovery range and soft checkpoint interval configuration parameter” in *Performance Guide*

log_to_redo_for_recovery - Amount of Log to be Redone for Recovery

Element identifier log_to_redo_for_recovery

Element type watermark

Table 354. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 355. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The amount of log (in bytes) that will have to be redone for crash recovery.

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.

Related reference:

- “log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages ” on page 294

log_write_time - Log Write Time

Element identifier log_write_time

Element type time

Table 356. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 357. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The total elapsed time spent by the logger writing log data to the disk.

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.

Database configuration monitor elements

Related reference:

- “log_writes - Number of Log Pages Written ” on page 291
- “num_log_write_io - Number of Log Writes ” on page 296

log_read_time - Log Read Time

Element identifier log_read_time

Element type time

Table 358. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 359. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The total elapsed time spent by the logger reading log data from the disk.

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.

Related reference:

- “log_reads - Number of Log Pages Read ” on page 291
- “num_log_data_found_in_buffer - Number of Log Data Found In Buffer ” on page 298
- “num_log_read_io - Number of Log Reads ” on page 297

num_log_write_io - Number of Log Writes

Element identifier num_log_write_io

Element type counter

Table 360. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 361. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The number of I/O requests issued by the logger for writing log data to the disk.

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.

Related reference:

- “log_write_time - Log Write Time ” on page 295
- “log_writes - Number of Log Pages Written ” on page 291

num_log_read_io - Number of Log Reads

Element identifier num_log_read_io

Element type counter

Table 362. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 363. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The number of I/O requests issued by the logger for reading log data from the disk.

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.

Related reference:

- “log_read_time - Log Read Time ” on page 296
- “log_reads - Number of Log Pages Read ” on page 291

num_log_part_page_io - Number of Partial Log Page Writes

Element identifier num_log_part_page_io

Element type counter

Table 364. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 365. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The number of I/O requests issued by the logger for writing partial log data to the disk.

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.

Related reference:

- “log_write_time - Log Write Time ” on page 295

Database configuration monitor elements

- “log_writes - Number of Log Pages Written ” on page 291
- “num_log_write_io - Number of Log Writes ” on page 296

num_log_buffer_full - Number of Full Log Buffers

Element identifier	num_log_buffer_full
Element type	counter

Table 366. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Description

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.

Usage Use this element to determine if the LOGBUFSZ database configuration parameter needs to be increased.

Related reference:

- “logbufsz - Log buffer size configuration parameter” in *Performance Guide*

num_log_data_found_in_buffer - Number of Log Data Found In Buffer

Element identifier	num_log_data_found_in_buffer
Element type	counter

Table 367. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 368. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

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.

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.

Related reference:

- “logbufsz - Log buffer size configuration parameter” in *Performance Guide*
- “num_log_read_io - Number of Log Reads ” on page 297

first_active_log - First Active Log File Number

Element identifier first_active_log

Element type information

Table 369. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	detail_log	Basic

Table 370. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The file number of the first active log file.

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.

Related reference:

- “current_active_log - Current Active Log File Number ” on page 300
- “last_active_log - Last Active Log File Number ” on page 299

last_active_log - Last Active Log File Number

Element identifier last_active_log

Element type information

Table 371. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	detail_log	Basic

Table 372. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The file number of the last active log file.

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.

Related reference:

- “current_active_log - Current Active Log File Number ” on page 300

Database configuration monitor elements

- “first_active_log - First Active Log File Number ” on page 299

current_active_log - Current Active Log File Number

Element identifier current_active_log

Element type information

Table 373. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	detail_log	Basic

Table 374. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

The file number of the active log file the DB2 database system is currently writing.

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.

Related reference:

- “first_active_log - First Active Log File Number ” on page 299
- “last_active_log - Last Active Log File Number ” on page 299

current_archive_log - Current Archive Log File Number

Element identifier current_archive_log

Element type information

Table 375. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	detail_log	Basic

Table 376. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Description

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`.

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

Database and application activity monitor elements

The following sections provide information on database and application activity.

- Locks and deadlocks monitor elements
- Lock wait information monitor elements
- Rollforward monitoring monitor elements
- Table space activity monitor elements
- Table activity monitor elements
- Table reorganization monitor elements
- SQL cursors monitor elements
- SQL statement activity monitor elements
- SQL statement details monitor elements
- Subsection details monitor elements
- Dynamic SQL monitor elements
- Intra-query parallelism monitor elements
- CPU usage monitor elements
- Snapshot monitoring monitor elements
- Event monitoring monitor elements

Locks and deadlocks

Locks and deadlocks monitor elements

The following elements provide information about locks and deadlocks:

- locks_held - Locks Held monitor element
- lock_list_in_use - Total Lock List Memory In Use monitor element
- deadlocks - Deadlocks Detected monitor element
- data_partition_id - Data Partition Identifier monitor element
data_partition_id - Data Partition Identifier monitor element
- lock_escals - Number of Lock Escalations monitor element
- x_lock_escals - Exclusive Lock Escalations monitor element
- lock_mode - Lock Mode monitor element
- lock_status - Lock Status monitor element
- lock_object_type - Lock Object Type Waited On monitor element
- lock_object_name - Lock Object Name monitor element
- lock_node - Lock Node monitor element
- lock_timeouts - Number of Lock Timeouts monitor element
- locks_held_top - Maximum Number of Locks Held monitor element
- dl_conns - Connections Involved in Deadlock monitor element
- lock_escalation - Lock Escalation monitor element
- lock_mode_requested - Lock Mode Requested monitor element
- deadlock_id - Deadlock Event Identifier monitor element
- deadlock_node - Partition Number Where Deadlock Occurred monitor element
- participant_no - Participant within Deadlock monitor element

Database and application activity monitor elements

- participant_no_holding_lk - Participant Holding a Lock on the Object Required by Application monitor element
- rolled_back_participant_no - Rolled Back Application Participant monitor element
- locks_in_list - Number of Locks Reported monitor element
- lock_name - Lock Name monitor element
- lock_attributes - Lock Attributes monitor element
- lock_release_flags - Lock Release Flags monitor element
- lock_count - Lock Count monitor element
- lock_hold_count - Lock Hold Count monitor element
- lock_current_mode - Original Lock Mode Before Conversion monitor element
- num_indoubt_trans - Number of Indoubt Transactions monitor element
elementnum_indoubt_trans - Number of Indoubt Transactions monitor element

locks_held - Locks Held

Element identifier locks_held

Element type gauge

Table 377. 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 378. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	event_detailed_dlconn	-

Description

The number of locks currently held.

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.

Related reference:

- “lock_escals - Number of Lock Escalations ” on page 304
- “x_lock_escals - Exclusive Lock Escalations ” on page 305
- “locks_held_top - Maximum Number of Locks Held ” on page 310

lock_list_in_use - Total Lock List Memory In Use

Element identifier lock_list_in_use

Element type water mark

Table 379. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The total amount of lock list memory (in bytes) that is in use.

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.

deadlocks - Deadlocks Detected

Element identifier deadlocks

Element type counter

Table 380. 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 381. Event Monitoring Information

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

Description

The total number of deadlocks that have occurred.

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.

Database and application activity monitor elements

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.

Related reference:

- “db_conn_time - Database Activation Timestamp ” on page 155
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “lock_escals - Number of Lock Escalations ” on page 304
- “x_lock_escals - Exclusive Lock Escalations ” on page 305
- “appl_id_holding_lk - Application ID Holding Lock ” on page 322

lock_escals - Number of Lock Escalations

Element identifier lock_escals

Element type counter

Table 382. 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 383. Event Monitoring Information

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

Description

The number of times that locks have been escalated from several row locks to a table lock.

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

Database and application activity monitor elements

- 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*.

Related reference:

- “*db_conn_time* - Database Activation Timestamp ” on page 155
- “*x_lock_escals* - Exclusive Lock Escalations ” on page 305
- “*locks_held_top* - Maximum Number of Locks Held ” on page 310

x_lock_escals - Exclusive Lock Escalations

Element identifier x_lock_escals

Element type counter

Table 384. 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 385. Event Monitoring Information

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

Description

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.

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

Database and application activity monitor elements

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.

Related reference:

- “db_conn_time - Database Activation Timestamp ” on page 155
- “lock_escals - Number of Lock Escalations ” on page 304
- “appl_con_time - Connection Request Start Timestamp ” on page 183
- “locks_held_top - Maximum Number of Locks Held ” on page 310

lock_mode - Lock Mode

Element identifier	lock_mode
Element type	information

Table 386. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	lock	Lock
Lock	lock_wait	Lock

Table 387. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

The type of lock being held.

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).

Database and application activity monitor elements

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

Element identifier	lock_status
Element type	information

Table 388. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic

Table 389. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-

Description

Indicates the internal status of the 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.

Related reference:

- “lock_mode - Lock Mode ” on page 306
- “lock_object_type - Lock Object Type Waited On ” on page 308
- “lock_object_name - Lock Object Name ” on page 308
- “table_file_id - Table File ID ” on page 360

lock_object_type - Lock Object Type Waited On

Element identifier lock_object_type

Element type information

Table 390. 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 391. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

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).

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

Element identifier lock_object_name

Element type information

Table 392. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	appl_lock_list	Lock
Lock	lock	Basic

Table 393. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

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).

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*.

Related reference:

- “lock_object_type - Lock Object Type Waited On ” on page 308
- “tablespace_name - Table Space Name ” on page 328
- “table_name - Table Name ” on page 351
- “table_schema - Table Schema Name ” on page 352

lock_node - Lock Node

Element identifier lock_node
Element type information

Table 394. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement
Deadlocks	event_dlconn	Statement
Deadlocks with Details	event_detailed_dlconn	Statement

Description

The node involved in a lock.

Usage This can be used for troubleshooting.

lock_timeouts - Number of Lock Timeouts

Element identifier lock_timeouts
Element type counter

Table 395. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic

Database and application activity monitor elements

For snapshot monitoring, this counter can be reset.

Table 396. Event Monitoring Information

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

Description

The number of times that a request to lock an object timed-out instead of being granted.

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

Element identifier locks_held_top

Element type counter

Table 397. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Transactions	event_xact	-

Description

The maximum number of locks held during this transaction.

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.

Related reference:

- “locks_held - Locks Held ” on page 302

- “lock_escals - Number of Lock Escalations ” on page 304
- “x_lock_escals - Exclusive Lock Escalations ” on page 305

dl_conns - Connections Involved in Deadlock

Element identifier dl_conns
 Element type gauge

Table 398. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-

Description

The number of connections that are involved in the 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

Element identifier lock_escalation
 Element type information

Table 399. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Lock
Lock	lock_wait	Lock

Table 400. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

Indicates whether a lock request was made as part of a lock escalation.

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

Element identifier lock_mode_requested
 Element type information

Table 401. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock_wait	Lock

Database and application activity monitor elements

Table 402. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

The lock mode being requested by the application.

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

Element identifier deadlock_id

Element type information

Table 403. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-
Deadlocks with Details History	event_detailed_dlconn	-
Deadlocks with Details History	event_stmt_history	-
Deadlocks with Details History Values	event_data_value	-
Deadlocks with Details History Values	event_detailed_dlconn	-
Deadlocks with Details History Values	event_stmt_history	-

Description

The deadlock identifier for a deadlock.

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

Element identifier deadlock_node

Element type information

Table 404. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

Partition number where the deadlock occurred.

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

Element identifier participant_no

Element type information

Table 405. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

A sequence number uniquely identifying this participant within this deadlock.

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

Element identifier participant_no_holding_lk

Element type information

Table 406. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

The participant number of the application that is holding a lock on the object that this application is waiting to obtain.

Usage This element can help you determine which applications are in contention for resources.

Related reference:

- “appl_id_holding_lk - Application ID Holding Lock ” on page 322

rolled_back_participant_no - Rolled Back Application Participant

Element identifier rolled_back_participant_no

Element type information

Table 407. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-

Description

The participant number identifying the rolled back application.

Database and application activity monitor elements

Usage A system administrator can use this information to determine which application did not complete its updates, and determine which application should be started.

Related reference:

- “rolled_back_appl_id - Rolled Back Application ” on page 323

locks_in_list - Number of Locks Reported

Element identifier locks_in_list

Element type information

Table 408. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	event_detailed_dlconn	-

Description

The number of locks held by a particular application to be reported on by the event monitor.

lock_name - Lock Name

Element identifier lock_name

Element type information

Table 409. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	lock_wait

Table 410. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-

Description

Internal binary lock name. This element serves as a unique identifier for locks.

lock_attributes - Lock Attributes

Element identifier lock_attributes

Element type information

Table 411. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	Basic

Table 412. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-

Description

Lock attributes.

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

Element identifier lock_release_flags

Element type information

Table 413. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	Basic

Table 414. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-

Description

Lock release flags.

Usage The following are possible release flag settings. Each release flag is based upon a bit flag value defined in sqlmon.h.

API Constant	Description
SQLM_LOCKRELFLAGS_SQLCOMPILER	Locks by SQL compiler.
SQLM_LOCKRELFLAGS_UNTRACKED	Non-unique, untracked locks.

Note: All non-assigned bits are used for application cursors.

Database and application activity monitor elements

lock_count - Lock Count

Element identifier lock_count

Element type gauge

Table 415. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic

Table 416. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-

Description

The number of locks on the lock being held.

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

Element identifier lock_hold_count

Element type gauge

Table 417. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic

Table 418. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-

Description

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.

lock_current_mode - Original Lock Mode Before Conversion

Element identifier lock_current_mode

Element type information

Table 419. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	Basic

Table 420. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-

Description

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.

Related concepts:

- “Lock conversion” in *Performance Guide*

num_indoubt_trans - Number of Indoubt Transactions

Element identifier num_indoubt_trans

Element type gauge

Table 421. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

The number of outstanding indoubt transactions in the database.

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

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 monitor element
- `lock_wait_time` - Time Waited On Locks monitor element
- `locks_waiting` - Current Agents Waiting On Locks monitor element
- `uow_lock_wait_time` - Total Time Unit of Work Waited on Locks monitor element

Database and application activity monitor elements

- lock_wait_start_time - Lock Wait Start Timestamp monitor element
- lock_timeout_val - Lock timeout monitor elementlock_timeout_val - Lock timeout monitor element
- agent_id_holding_lock - Agent ID Holding Lock monitor element
- appl_id_holding_lk - Application ID Holding Lock monitor element
- sequence_no_holding_lk - Sequence Number Holding Lock monitor element
- rolled_back_appl_id - Rolled Back Application monitor element
- rolled_back_agent_id - Rolled Back Agent monitor element
- rolled_back_sequence_no - Rolled Back Sequence Number monitor element
- data_partition_id - Data Partition Identifier monitor element

lock_waits - Lock Waits

Element identifier lock_waits

Element type counter

Table 422. 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 423. Event Monitoring Information

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

Description

The total number of times that applications or connections waited for locks.

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.

Related concepts:

- “Locks and concurrency control” in *Performance Guide*

Related reference:

- “appl_con_time - Connection Request Start Timestamp ” on page 183

- “lock_wait_time - Time Waited On Locks ” on page 319

lock_wait_time - Time Waited On Locks

Element identifier lock_wait_time

Element type counter

Table 424. 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 425. Event Monitoring Information

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

Description

The total elapsed time waited for a lock. Elapsed time is given in milliseconds.

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.

Related reference:

- “locks_waiting - Current Agents Waiting On Locks ” on page 320
- “lock_waits - Lock Waits ” on page 318

locks_waiting - Current Agents Waiting On Locks

Element identifier locks_waiting

Element type gauge

Table 426. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Lock	db_lock_list	Basic

Description

Indicates the number of agents waiting on a lock.

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.

Related reference:

- “*appls_cur_cons* - Applications Connected Currently ” on page 194

uow_lock_wait_time - Total Time Unit of Work Waited on Locks

Element identifier uow_lock_wait_time

Element type counter

Table 427. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Unit of Work

Description

The total amount of elapsed time this unit of work has spent waiting for locks.

Usage This element can help you determine the severity of the resource contention problem.

lock_wait_start_time - Lock Wait Start Timestamp

Element identifier lock_wait_start_time

Element type timestamp

Table 428. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock, Timestamp
Lock	lock_wait	Lock, Timestamp

Table 429. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_dlconn	Timestamp
Deadlocks with Details	event_detailed_dlconn	Timestamp

Description

The date and time that this application started waiting to obtain a lock on the object that is currently locked by another application.

Usage This element can help you determine the severity of resource contention.

Related reference:

- “agent_id_holding_lock - Agent ID Holding Lock ” on page 321

lock_timeout_val - Lock timeout

Element identifier lock_timeout_val

Element type information

Table 430. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic
Application	agent	Basic

Description

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.

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

Element identifier agent_id_holding_lock

Element type information

Table 431. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	appl_lock_list	Lock
Lock	lock_wait	Lock

Description

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.

Database and application activity monitor elements

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.

Related reference:

- “`lock_wait_start_time` - Lock Wait Start Timestamp ” on page 320
- “`appl_id_holding_lk` - Application ID Holding Lock ” on page 322

appl_id_holding_lk - Application ID Holding Lock

Element identifier `appl_id_holding_lk`

Element type information

Table 432. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	appl_lock_list	Lock
Lock	lock_wait	Lock

Table 433. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

The application ID of the application that is holding a lock on the object that this application is waiting to obtain.

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.

Related reference:

- “agent_id_holding_lock - Agent ID Holding Lock ” on page 321
- “deadlocks - Deadlocks Detected ” on page 303

sequence_no_holding_lk - Sequence Number Holding Lock

Element identifier sequence_no_holding_lk
Element type information

Table 434. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic
Lock	appl_lock_list	Basic

Table 435. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

The sequence number of the application that is holding a lock on the object that this application is waiting to obtain.

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

Element identifier rolled_back_appl_id
Element type information

Table 436. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-

Description

Application id that was rolled back when a deadlock occurred.

Usage A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted.

Related reference:

- “coord_agents_top - Maximum Number of Coordinating Agents ” on page 199

rolled_back_agent_id - Rolled Back Agent

Element identifier rolled_back_agent_id
Element type information

Database and application activity monitor elements

Table 437. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-

Description

Agent that was rolled back when a deadlock occurred.

Usage A system administrator can use this information to determine which application did not complete its updates, and determine which applications should be restarted.

Related reference:

- “coord_agents_top - Maximum Number of Coordinating Agents ” on page 199

rolled_back_sequence_no - Rolled Back Sequence Number

Element identifier rolled_back_sequence_no

Element type information

Table 438. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-

Description

The sequence number of the application that was rolled back when a deadlock occurred.

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

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 monitor element
- ts_name - Tablespace Being Rolled Forward monitor element
- rf_type - Rollforward Type monitor element
- rf_log_num - Log Being Rolled Forward monitor element
- rf_status - Log Phase monitor element

Progress monitoring of the runtime rollback process

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

Unit of work rollback

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

Savepoint rollback

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

Database and application activity monitor elements

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

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

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

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

Sample output from GET SNAPSHOT FOR ALL APPLICATIONS command:

Application Snapshot

```
Application handle      = 6
Application status     = Rollback Active
  Start Time           = 02/20/2004 12:49:27.713720
  Completed Work       = 1024000 bytes
  Total Work           = 4084000 bytes
```

Application Snapshot

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

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

rf_timestamp - Rollforward Timestamp

Element identifier rf_timestamp

Element type timestamp

Table 439. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollforward	Timestamp

Description

The timestamp of the last committed transaction..

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.

Related reference:

- “ts_name - Tablespace Being Rolled Forward ” on page 325

ts_name - Tablespace Being Rolled Forward

Element identifier ts_name

Element type information

Database and application activity monitor elements

Table 440. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollforward	Basic

Description

The name of the table space currently rolled forward.

Usage If a rollforward is in progress, this element identifies the table spaces involved.

Related reference:

- “rf_timestamp - Rollforward Timestamp ” on page 325

rf_type - Rollforward Type

Element identifier rf_type

Element type information

Table 441. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollforward	Basic

Description

The type of rollforward in progress.

Usage An indicator of whether recovery is happening at a database or table space level.

rf_log_num - Log Being Rolled Forward

Element identifier rf_log_num

Element type information

Table 442. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollforward	Basic

Description

The log being processed.

Usage If a rollforward is in progress, this element identifies the log involved.

rf_status - Log Phase

Element identifier rf_status

Element type information

Table 443. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollforward	Basic

Description

The status of the recovery.

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

Table space activity monitor elements

The following elements provide information about the table spaces:

- `tablespace_id` - Table Space Identification monitor element
- `tablespace_name` - Table Space Name monitor element
- `tablespace_type` - Table Space Type monitor element
- `tablespace_content_type` - Table Space Contents Type monitor element
- `tablespace_state` - Table Space State monitor element
- `tablespace_page_size` - Table Space Page Size monitor element
- `tablespace_extent_size` - Table Space Extent Size monitor element
- `tablespace_prefetch_size` - Table Space Prefetch Size monitor element
- `tablespace_cur_pool_id` - Buffer Pool Currently Being Used monitor element
- `tablespace_next_pool_id` - Buffer Pool That Will Be Used at Next Startup monitor element
- `tablespace_total_pages` - Total Pages in Table Space monitor element
- `tablespace_usable_pages` - Usable Pages in Table Space monitor element
- `tablespace_used_pages` - Used Pages in Table Space monitor element
- `tablespace_free_pages` - Free Pages in Table Space monitor element
- `tablespace_pending_free_pages` - Pending Free Pages in Table Space monitor element
- `tablespace_page_top` - Table Space High Water Mark monitor element
- `tablespace_current_size` - Current table space size monitor element
`tablespace_current_size` - Current table space size monitor element
- `tablespace_initial_size` - Initial table space size monitor element
`tablespace_initial_size` - Initial table space size monitor element
- `tablespace_max_size` - Maximum table space size monitor element
`tablespace_max_size` - Maximum table space size monitor element
- `tablespace_increase_size` - Increase size in bytes monitor element
`tablespace_increase_size` - Increase size in bytes monitor element
- `tablespace_increase_size_percent` - Increase size by percent monitor element
`tablespace_increase_size_percent` - Increase size by percent monitor element
- `tablespace_last_resize_time` - Time of last successful resize monitor element
`tablespace_last_resize_time` - Time of last successful resize monitor element
- `tablespace_rebalancer_mode` - Rebalancer Mode monitor element
- `tablespace_rebalancer_start_time` - Rebalancer Start Time monitor element
- `tablespace_rebalancer_restart_time` - Rebalancer Restart Time monitor element
- `tablespace_using_auto_storage` - Using automatic storage monitor element
`tablespace_using_auto_storage` - Using automatic storage monitor element
- `tablespace_auto_resize_enabled` - Auto-resize enabled monitor element
`tablespace_auto_resize_enabled` - Auto-resize enabled monitor element

Database and application activity monitor elements

- `tablespace_last_resize_failed` - Last resize attempt failed monitor element
- `tablespace_rebalancer_extents_remaining` - Total Number of Extents to be Processed by the Rebalancer monitor element
- `tablespace_rebalancer_extents_processed` - Number of Extents the Rebalancer has Processed monitor element
- `tablespace_rebalancer_last_extent_moved` - Last Extent Moved by the Rebalancer monitor element
- `tablespace_rebalancer_priority` - Current Rebalancer Priority monitor element
- `tablespace_num_quiescers` - Number of Quiescers monitor element
- `fs_caching` - File System Caching monitor element
- `tablespace_state_change_object_id` - State Change Object Identification monitor element
- `tablespace_state_change_ts_id` - State Change Table Space Identification monitor element
- `tablespace_min_recovery_time` - Minimum Recovery Time For Rollforward monitor element
- `tablespace_num_containers` - Number of Containers in Table Space monitor element
- Container status monitor elements
- `tablespace_num_ranges` - Number of Ranges in the Table Space Map monitor element
- Table space range status monitor elements

tablespace_id - Table Space Identification

Element identifier `tablespace_id`

Element type information

Table 444. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic
Table	table	Basic

Table 445. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

An integer that uniquely represents a table space used by the current database.

Usage The value of this element matches a value from column `TBSPACEID` of view `SYSCAT.TABLESPACES`.

tablespace_name - Table Space Name

Element identifier `tablespace_name`

Element type information

Table 446. 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 447. 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	-

Description

The name of a table space.

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.

Related reference:

- “lock_object_type - Lock Object Type Waited On ” on page 308

tablespace_type - Table Space Type

Element identifier tablespace_type

Element type information

Table 448. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

The type of a table space.

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:

Database and application activity monitor elements

- For DMS: SQLM_TABLESPACE_TYP_DMS
- For SMS: SQLM_TABLESPACE_TYP_SMS

tablespace_content_type - Table Space Contents Type

Element identifier tablespace_content_type
Element type information

Table 449. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

The type of content in a table space.

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

Element identifier tablespace_state
Element type information

Table 450. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

This element describes the current state of a table space.

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 451. 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

Database and application activity monitor elements

Table 451. Bit definitions listed in sqlutil.h (continued)

Hexadecimal Value	Decimal Value	State
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

Element identifier tablespace_page_size

Element type information

Table 452. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

Page size used by a table space in bytes.

tablespace_extent_size - Table Space Extent Size

Element identifier tablespace_extent_size

Element type information

Table 453. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

The extent size used by a table space.

tablespace_prefetch_size - Table Space Prefetch Size

Element identifier tablespace_prefetch_size

Element type information

Database and application activity monitor elements

Table 454. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic
Table Space	tablespace_nodeinfo	Basic

Description

The maximum number of pages the prefetcher gets from the disk at a time.

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

Element identifier tablespace_cur_pool_id

Element type information

Table 455. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

The buffer pool identifier for a buffer pool that a table space is currently using.

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

Element identifier tablespace_next_pool_id

Element type information

Table 456. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

The buffer pool identifier for a buffer pool that a table space will use at the next database startup.

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

Element identifier tablespace_total_pages

Element type information

Table 457. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic (DMS table spaces) Buffer Pool (SMS table spaces)

Description

Total number of pages in a table space.

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

Element identifier tablespace_usable_pages

Element type information

Table 458. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic (DMS table spaces) Buffer Pool (SMS table spaces)

Description

The total number of pages in a table space minus overhead pages.

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

Element identifier tablespace_used_pages

Element type information

Table 459. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic (DMS table spaces) Buffer Pool (SMS table spaces)

Description

The total number of pages that are currently used (not free) in a table space.

Database and application activity monitor elements

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

Element identifier `tablespace_free_pages`

Element type `information`

Table 460. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	<code>tablespace_nodeinfo</code>	Basic

Description

The total number of pages that are currently free in a table space.

Usage This is applicable only to a DMS table space.

tablespace_pending_free_pages - Pending Free Pages in Table Space

Element identifier `tablespace_pending_free_pages`

Element type `information`

Table 461. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	<code>tablespace_nodeinfo</code>	Basic

Description

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.

Usage This is applicable only to a DMS table space.

tablespace_page_top - Table Space High Water Mark

Element identifier `tablespace_page_top`

Element type `information`

Table 462. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	<code>tablespace_nodeinfo</code>	Basic

Description

The page in a table space that is holding the high-water mark.

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 water mark", but rather a "current water mark", since the value can decrease. For SMS, this is not applicable.

tablespace_using_auto_storage - Using automatic storage

Element identifier `tablespace_using_auto_storage`

Element type `information`

Table 463. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

This element describes whether the table space was created as an automatic storage table space. A value of 1 means yes; 0 means no.

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

Element identifier tablespace_auto_resize_enabled
Element type information

Table 464. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

This element describes whether automatic resizing is enabled for the table space. A value of 1 means yes; 0 means no.

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

Element identifier tablespace_initial_size
Element type information

Table 465. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The initial size of the automatic storage table space in bytes.

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

Element identifier tablespace_current_size
Element type information

Database and application activity monitor elements

Table 466. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

This element shows the current size of the table space in bytes.

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

Element identifier tablespace_max_size

Element type information

Table 467. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

This element shows the maximum size in bytes to which the table space can automatically resize or increase.

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

Element identifier tablespace_increase_size

Element type information

Table 468. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

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.

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

Element identifier tablespace_increase_size_percent

Element type information

Table 469. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

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.

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

Element identifier tablespace_last_resize_time

Element type information

Table 470. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

This element shows a timestamp representing the last time that the size of the table space was successfully increased.

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

Element identifier tablespace_last_resize_failed

Database and application activity monitor elements

Element type information

Table 471. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

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.

Usage For an automatic storage table space, this element may show that there is no space left on any of the database's storage paths. For a non-automatic storage table space, a failure means that one of the containers could not be extended because its filesystem was full. Another reason for failure is that the maximum size of the table space has been reached. This element is only applicable to table spaces that are enabled to be automatically resized.

tablespace_rebalancer_mode - Rebalancer Mode

Element identifier tablespace_rebalancer_mode

Element type information

Table 472. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

An integer that represents whether a forward or reverse rebalance is taking place. Its 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

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

Element identifier tablespace_rebalancer_start_time

Element type information

Table 473. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

A timestamp representing when a rebalancer was initially started.

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

Element identifier tablespace_rebalancer_restart_time

Element type information

Table 474. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

A timestamp representing when a rebalancer was restarted after being paused or stopped.

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

Element identifier tablespace_rebalancer_extents_remaining

Element type information

Table 475. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The number of extents to be moved. This value is calculated at either the rebalancer start time or restart time (whichever is most recent).

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

Element identifier tablespace_rebalancer_extents_processed

Element type information

Table 476. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The number of extents that the rebalancer has already moved since the rebalancer has been started or restarted (whichever is most recent).

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

Element identifier tablespace_rebalancer_last_extent_moved
Element type information

Table 477. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The last extent moved by the rebalancer.

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

Element identifier tablespace_rebalancer_priority
Element type information

Table 478. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The priority at which the rebalancer is running in the database.

Usage This is only applicable to a DMS table space.

tablespace_num_quiescers - Number of Quiescers

Element identifier tablespace_num_quiescers
Element type information

Table 479. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The number of users quiescing the table space (can be in the range of 0 to 5).

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

Table space quiescer activity monitor elements

Table space quiescer activity monitor elements: The following elements provide information about table space quiescer activity:

- quiescer_auth_id - Quiescer User Authorization Identification monitor element
- quiescer_agent_id - Quiescer Agent Identification monitor element
- quiescer_ts_id - Quiescer Table Space Identification monitor element
- quiescer_obj_id - Quiescer Object Identification monitor element
- quiescer_state - Quiescer State monitor element

quiescer_auth_id - Quiescer User Authorization Identification :

Element identifier quiescer_auth_id

Element type information

Table 480. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_quiescer	Basic

Description

Authorization ID of the user holding a quiesce state.

Usage Use this element to determine who is responsible for quiescing a table space.

quiescer_agent_id - Quiescer Agent Identification :

Element identifier quiescer_agent_id

Element type information

Table 481. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_quiescer	Basic

Description

Agent ID of the agent holding a quiesce state.

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 :

Element identifier quiescer_ts_id

Element type information

Table 482. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_quiescer	Basic

Database and application activity monitor elements

Description

The table space ID of the object that causes a table space to be quiesced.

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 :

Element identifier `quiescer_obj_id`

Element type information

Table 483. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_quiescer	Basic

Description

The object ID of the object that causes a table space to be quiesced.

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 :

Element identifier `quiescer_state`

Element type information

Table 484. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_quiescer	Basic

Description

The type of quiesce being done (for example, "SHARE", "INTENT TO UPDATE", or "EXCLUSIVE").

Usage The value of this element matches the value of constants `SQLB QUIESCED SHARE`, `SQLB QUIESCED UPDATE`, or `SQLB QUIESCED EXCLUSIVE` from `sqlutil.h`.

`tablespace_state_change_object_id` - State Change Object Identification

Element identifier `tablespace_state_change_object_id`

Element type information

Table 485. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The object that caused the table space state to be set to "Load pending" or "Delete pending".

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

Element identifier tablespace_state_change_ts_id
Element type information

Table 486. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

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.

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

Element identifier tablespace_min_recovery_time
Element type information

Table 487. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

A timestamp showing the earliest point in time to which a table space can be rolled forward.

Usage Displayed only if non zero.

tablespace_num_containers - Number of Containers in Table Space

Element identifier tablespace_num_containers
Element type information

Table 488. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

Total number of containers in the table space.

Container status

Container status monitor elements: The following elements provide information about container status:

- container_id - Container Identification monitor element

Database and application activity monitor elements

- container_name - Container Name monitor element
- container_type - Container Type monitor element
- container_total_pages - Total Pages in Container monitor element
- container_usable_pages - Usable Pages in Container monitor element
- container_stripe_set - Stripe Set monitor element
- container_accessible - Accessibility of Container monitor element

container_id - Container Identification :

Element identifier container_id
Element type information

Table 489. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_container	Basic

Description

An integer that uniquely defines a container within a table space.

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 :

Element identifier container_name
Element type information

Table 490. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_container	Basic

Description

The name of a container.

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 :

Element identifier container_type
Element type information

Table 491. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_container	Basic

Description

The type of the container.

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 :

Element identifier container_total_pages

Element type information

Table 492. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_container	Basic (DMS table spaces) Buffer Pool (SMS table spaces)

Description

The total number of pages occupied by the container.

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 :

Element identifier container_usable_pages

Element type information

Table 493. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_container	Basic (DMS table spaces) Buffer Pool (SMS table spaces)

Description

The total number of usable pages in a container.

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 :

Element identifier container_stripe_set

Element type information

Database and application activity monitor elements

Table 494. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_container	Basic

Description

The stripe set that a container belongs to.

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 :

Element identifier container_accessible

Element type information

Table 495. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_container	Basic

Description

This element describes if a container is accessible or not (1 meaning yes, 0 meaning no).

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.

tablespace_num_ranges - Number of Ranges in the Table Space Map

Element identifier tablespace_num_ranges

Element type information

Table 496. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_nodeinfo	Basic

Description

The number of ranges (entries) in the table space map. This can be in the range of 1 to 100's (but is usually less than a dozen). The table space map only exists for DMS table spaces.

Table space range status

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)

Database and application activity monitor elements

Stripe	Range	MaxPage	MaxExtent	StartStripe	EndStripe	Adj	# Confs	Containers
1	[1]	999	499	125	249	0	3	(0,1,2)
2	[2]	1499	749	250	374	0	1	(1,2)

For each range, the following information will be returned in the snapshot (templates follow):

- range_stripe_set_number - Stripe Set Number monitor element
- range_number - Range Number monitor element
- range_max_page_number - Maximum Page in Range monitor element
- range_max_extent - Maximum Extent in Range monitor element
- range_start_stripe - Start Stripe monitor element
- range_end_stripe - End Stripe monitor element
- range_adjustment - Range Adjustment monitor element
- range_num_containers - Number of Containers in Range monitor element
- 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.
 - range_container_id - Range Container monitor element
 - range_offset - Range Offset monitor element

range_stripe_set_number - Stripe Set Number :

Element identifier range_stripe_set_number

Element type information

Table 497. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the stripe set in which a range resides.

Usage This element is applicable only to a DMS table space.

range_number - Range Number :

Element identifier range_number

Element type information

Table 498. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the number of a range within the table space map.

Usage This element is applicable only to a DMS table space.

range_max_page_number - Maximum Page in Range :

Element identifier range_max_page_number

Element type information

Database and application activity monitor elements

Table 499. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the maximum page number that is mapped by a range.

Usage This element is applicable only to a DMS table space.

range_max_extent - Maximum Extent in Range :

Element identifier range_max_extent

Element type information

Table 500. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the maximum extent number that is mapped by a range.

Usage This element is applicable only to a DMS table space.

range_start_stripe - Start Stripe :

Element identifier range_start_stripe

Element type information

Table 501. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the number of the first stripe in a range.

Usage This element is applicable only to a DMS table space.

range_end_stripe - End Stripe :

Element identifier range_end_stripe

Element type information

Table 502. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the number of the last stripe in a range.

Usage This element is applicable only to a DMS table space.

range_adjustment - Range Adjustment :

Element identifier range_adjustment

Element type information

Table 503. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the offset into the container array in which a range actually starts.

Usage This element is applicable only to a DMS table space.

range_num_containers - Number of Containers in Range :

Element identifier range_num_containers

Element type information

Table 504. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

This value represents the number of containers in the current range.

Usage This element is applicable only to a DMS table space.

range_container_id - Range Container :

Element identifier range_container_id

Element type information

Table 505. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

An integer that uniquely defines a container within a range.

Usage This element is applicable only to a DMS table space.

range_offset - Range Offset :

Element identifier range_offset

Element type information

Table 506. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_range	Basic

Description

The offset from stripe 0 of the beginning of the stripe set to which a range belongs.

Usage This element is applicable only to a DMS table space.

Database and application activity monitor elements

fs_caching - File System Caching

Element identifier	fs_caching
Element type	information

Table 507. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table space	tablespace	Basic

Table 508. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tablespaces	event_tablespace	-

Description

Indicates whether a particular tablespace uses file system caching.

Table activity

Table activity monitor elements

The following elements provide information about the tables:

- table_type - Table Type monitor element
- table_name - Table Name monitor element
- table_schema - Table Schema Name monitor element
- rows_deleted - Rows Deleted monitor element
- rows_inserted - Rows Inserted monitor element
- rows_updated - Rows Updated monitor element
- rows_selected - Rows Selected monitor element
- rows_written - Rows Written monitor element
- rows_read - Rows Read monitor element
- overflow_accesses - Accesses to Overflowed Records monitor element
- int_rows_deleted - Internal Rows Deleted monitor element
- int_rows_updated - Internal Rows Updated monitor element
- int_rows_inserted - Internal Rows Inserted monitor element
- table_file_id - Table File ID monitor element
- page_reorgs - Page Reorganizations monitor element
- data_object_pages - Data Object Pages monitor element
data_object_pages - Data Object Pages monitor element
- index_object_pages - Index Object Pages monitor element
index_object_pages - Index Object Pages monitor element
- lob_object_pages - LOB Object Pages monitor element
lob_object_pages - LOB Object Pages monitor element
- long_object_pages - Long Object Pages monitor element
long_object_pages - Long Object Pages monitor element
- data_partition_id - Data Partition Identifier monitor element
data_partition_id - Data Partition Identifier monitor element

table_type - Table Type

Element identifier	table_type
Element type	information

Table 509. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Basic

Table 510. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The type of table for which information is returned.

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.

Related reference:

- “table_file_id - Table File ID ” on page 360

table_name - Table Name

Element identifier	table_name
Element type	information

Table 511. 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 512. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-
Deadlocks	lock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

The name of the table.

Usage Along with *table_schema*, this element can help you determine the source of contention for resources.

Database and application activity monitor elements

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

Related reference:

- “table_schema - Table Schema Name ” on page 352
- “lock_object_type - Lock Object Type Waited On ” on page 308
- “lock_object_name - Lock Object Name ” on page 308

table_schema - Table Schema Name

Element identifier table_schema

Element type information

Table 513. 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 514. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-
Deadlocks	lock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description

The schema of the table.

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

Related reference:

- “table_name - Table Name ” on page 351
- “lock_object_type - Lock Object Type Waited On ” on page 308
- “lock_object_name - Lock Object Name ” on page 308

rows_deleted - Rows Deleted

Element identifier rows_deleted
Element type counter

Table 515. 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 516. Event Monitoring Information

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

Description

This is the number of row deletions attempted.

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*.

Related reference:

- “int_rows_deleted - Internal Rows Deleted ” on page 358

rows_inserted - Rows Inserted

Element identifier rows_inserted

Database and application activity monitor elements

Element type counter

Table 517. 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 518. Event Monitoring Information

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

Description

This is the number of row insertions attempted.

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

Element identifier rows_updated

Element type counter

Table 519. 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 520. Event Monitoring Information

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

Description

This is the number of row updates attempted.

Usage You can use this element to gain insight into the current level of activity within the database.

Database and application activity monitor elements

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.

Related reference:

- “int_rows_updated - Internal Rows Updated ” on page 358

rows_selected - Rows Selected

Element identifier rows_selected

Element type counter

Table 521. 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 522. Event Monitoring Information

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

Description

This is the number of rows that have been selected and returned to the application.

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.

Related reference:

- “select_sql_stmts - Select SQL Statements Executed ” on page 377

rows_written - Rows Written

Element identifier rows_written

Element type counter

Database and application activity monitor elements

Table 523. 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 524. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-
Tables	event_table	-
Statements	event_stmt	-
Transactions	event_xact	-

Description

This is the number of rows changed (inserted, deleted or updated) in the table.

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.

Related reference:

- “rows_read - Rows Read ” on page 356
- “int_rows_inserted - Internal Rows Inserted ” on page 359
- “int_rows_deleted - Internal Rows Deleted ” on page 358
- “int_rows_updated - Internal Rows Updated ” on page 358

rows_read - Rows Read

Element identifier rows_read

Element type counter

Table 525. 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 526. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-
Tables	event_table	-
Statements	event_stmt	-
Transactions	event_xact	-

Description

This is the number of rows read from the table.

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.

Related reference:

- “rows_written - Rows Written ” on page 355
- “overflow_accesses - Accesses to Overflowed Records ” on page 357

overflow_accesses - Accesses to Overflowed Records

Element identifier overflow_accesses

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	-

Description

The number of accesses (reads and writes) to overflowed rows of this table.

Usage Overflowed rows indicate that data fragmentation has occurred. If this

Database and application activity monitor elements

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.

Related reference:

- “rows_written - Rows Written ” on page 355
- “rows_read - Rows Read ” on page 356

int_rows_deleted - Internal Rows Deleted

Element identifier int_rows_deleted

Element type counter

Table 529. 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 530. Event Monitoring Information

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

Description

This is the number of rows deleted from the database as a result of internal activity.

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.

Related reference:

- “rows_deleted - Rows Deleted ” on page 353

int_rows_updated - Internal Rows Updated

Element identifier int_rows_updated

Element type counter

Table 531. 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 532. Event Monitoring Information

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

Description

This is the number of rows updated from the database as a result of internal activity.

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.

Related reference:

- “rows_updated - Rows Updated ” on page 354

int_rows_inserted - Internal Rows Inserted

Element identifier int_rows_inserted

Element type counter

Table 533. 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 534. Event Monitoring Information

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

Database and application activity monitor elements

Description

The number of rows inserted into the database as a result of internal activity caused by triggers.

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.

Related reference:

- “rows_inserted - Rows Inserted ” on page 353

table_file_id - Table File ID

Element identifier table_file_id

Element type information

Table 535. 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 536. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-

Description

This is the file ID (FID) for the table.

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.

Related reference:

- “table_name - Table Name ” on page 351
- “table_schema - Table Schema Name ” on page 352
- “table_type - Table Type ” on page 350

page_reorgs - Page Reorganizations

Element identifier page_reorgs

Element type counter

Table 537. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Basic

For snapshot monitoring, this counter can be reset.

Table 538. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The number of page reorganizations executed for a 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.

Related reference:

- “rows_inserted - Rows Inserted ” on page 353
- “rows_updated - Rows Updated ” on page 354

data_object_pages - Data Object Pages

Element identifier data_object_pages

Element type information

Table 539. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Basic

Table 540. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The number of disk pages consumed by a table. This size represents the

Database and application activity monitor elements

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.

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.

Related reference:

- “*index_object_pages* - Index Object Pages ” on page 362
- “*lob_object_pages* - LOB Object Pages ” on page 362
- “*long_object_pages* - Long Object Pages ” on page 363
- “*xda_object_pages* - XDA Object Pages ” on page 267

index_object_pages - Index Object Pages

Element identifier index_object_pages

Element type information

Table 541. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Basic

Table 542. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The number of disk pages consumed by all indexes defined on a 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

Element identifier lob_object_pages

Element type information

Table 543. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Basic

Table 544. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The number of disk pages consumed by LOB data.

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

Element identifier long_object_pages
Element type information

Table 545. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Basic

Table 546. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The number of disk pages consumed by long data in a 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.

Table reorganization

Table reorganization monitor elements

The following elements provide information about table reorganization:

- reorg_type - Table Reorganize Attributes monitor element
- reorg_status - Table Reorganize Status monitor element
- reorg_phase - Reorganize Phase monitor element
- reorg_phase_start - Reorganize Phase Start Time monitor element
- reorg_max_phase - Maximum Reorganize Phase monitor element
- reorg_current_counter - Reorganize Progress monitor element
- reorg_max_counter - Total Amount of Reorganization monitor element
- reorg_completion - Reorganization Completion Flag monitor element
- reorg_start - Table Reorganize Start Time monitor element
- reorg_end - Table Reorganize End Time monitor element
- reorg_index_id - Index Used to Reorganize the Table monitor element
- reorg_tbspc_id - Table Space Where Table or Data partition is Reorganized monitor element
- reorg_rows_compressed - Rows Compressed monitor element
 elementreorg_rows_compressed - Rows Compressed monitor element
- reorg_rows_rejected_for_compression - Rows Rejected for Compression monitor element
 elementreorg_rows_rejected_for_compression - Rows Rejected for Compression monitor element
- data_partition_id - Data Partition Identifier monitor element
 data_partition_id - Data Partition Identifier monitor element

reorg_type - Table Reorganize Attributes

Element identifier reorg_type
Element type information

Database and application activity monitor elements

Table 547. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

Table reorganize attribute settings.

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_NOTTRUNCATE_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

Element identifier reorg_status

Element type information

Table 548. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The status of an in-place (online) table or a data partition level reorganization. This is not applicable to classic (offline) table reorganizations.

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

Element identifier reorg_phase
 Element type information

Table 549. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

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.

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

Element identifier reorg_phase_start
 Element type timestamp

Table 550. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

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.

reorg_max_phase - Maximum Reorganize Phase

Element identifier reorg_max_phase
 Element type information

Database and application activity monitor elements

Table 551. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

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]).

reorg_current_counter - Reorganize Progress

Element identifier reorg_current_counter

Element type information

Table 552. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

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. 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

Element identifier reorg_max_counter

Element type information

Table 553. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

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.

reorg_completion - Reorganization Completion Flag

Element identifier reorg_completion

Element type information

Table 554. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

Table reorganization success indicator. For partitioned tables, this will also indicate the completion status for the data partition.

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

Element identifier reorg_start

Element type timestamp

Table 555. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The start time of a table reorganization. For partitioned tables, this will also indicate the start time for each data partition reorganization.

reorg_end - Table Reorganize End Time

Element identifier reorg_end

Element type timestamp

Table 556. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The end time of a table reorganization. For partitioned tables, this will also indicate the end time for each data partition reorganization.

reorg_index_id - Index Used to Reorganize the Table

Element identifier reorg_index_id

Element type information

Table 557. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The index being used to reorganize the table.

reorg_tbspc_id - Table Space Where Table or Data partition is Reorganized

Element identifier reorg_tbspc_id

Database and application activity monitor elements

Element type information

Table 558. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The table space in which the table will be reorganized. For partitioned tables, this indicates the table space where each data partition is reorganized.

reorg_rows_compressed - Rows Compressed

Element identifier reorg_rows_compressed

Element type information

Table 559. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

Number of rows compressed in the table during reorganization.

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

Element identifier reorg_rows_rejected_for_compression

Element type information

Table 560. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

Number of rows that were not compressed during reorganization due to the record length being less than or equal to the minimum record length.

Usage A record will not be compressed if it is less than or equal to the minimum record length. The number of rows rejected reflects a running count for these records that fail to meet this compression requirement.

reorg_long_tbspc_id - Table Space Where Long Objects are Reorganized monitor element

Element identifier reorg_long_tbspc_id

Element type information

Table 561. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

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.

SQL cursors

SQL cursors monitor elements

The following elements provide information about the SQL cursors:

- open_rem_curs - Open Remote Cursors monitor element
- open_rem_curs_blk - Open Remote Cursors with Blocking monitor element
- rej_curs_blk - Rejected Block Cursor Requests monitor element
- acc_curs_blk - Accepted Block Cursor Requests monitor element
- open_loc_curs - Open Local Cursors monitor element
- open_loc_curs_blk - Open Local Cursors with Blocking monitor element

open_rem_curs - Open Remote Cursors

Element identifier open_rem_curs

Element type gauge

Table 562. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Description

The number of remote cursors currently open for this application, including those cursors counted by *open_rem_curs_blk*.

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*.

Related reference:

- “open_rem_curs_blk - Open Remote Cursors with Blocking ” on page 369
- “open_loc_curs - Open Local Cursors ” on page 371

open_rem_curs_blk - Open Remote Cursors with Blocking

Element identifier open_rem_curs_blk

Element type gauge

Database and application activity monitor elements

Table 563. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Description

The number of remote blocking cursors currently open for this application.

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*.

Related reference:

- “open_rem_curs - Open Remote Cursors ” on page 369
- “rej_curs_blk - Rejected Block Cursor Requests ” on page 370
- “acc_curs_blk - Accepted Block Cursor Requests ” on page 371
- “open_loc_curs - Open Local Cursors ” on page 371
- “open_loc_curs_blk - Open Local Cursors with Blocking ” on page 372

rej_curs_blk - Rejected Block Cursor Requests

Element identifier *rej_curs_blk*

Element type counter

Table 564. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 565. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

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.

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.

Related reference:

- “acc_curs_blk - Accepted Block Cursor Requests ” on page 371
- “open_loc_curs - Open Local Cursors ” on page 371
- “open_loc_curs_blk - Open Local Cursors with Blocking ” on page 372
- “open_rem_curs - Open Remote Cursors ” on page 369
- “open_rem_curs_blk - Open Remote Cursors with Blocking ” on page 369

acc_curs_blk - Accepted Block Cursor Requests

Element identifier acc_curs_blk

Element type counter

Table 566. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 567. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

The number of times that a request for an I/O block was accepted.

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.

Related reference:

- “rej_curs_blk - Rejected Block Cursor Requests ” on page 370
- “open_loc_curs - Open Local Cursors ” on page 371
- “open_loc_curs_blk - Open Local Cursors with Blocking ” on page 372
- “open_rem_curs - Open Remote Cursors ” on page 369
- “open_rem_curs_blk - Open Remote Cursors with Blocking ” on page 369

open_loc_curs - Open Local Cursors

Element identifier open_loc_curs

Element type gauge

Table 568. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Description

The number of local cursors currently open for this application, including those cursors counted by *open_loc_curs_blk*.

Usage You may use this element in conjunction with *open_loc_curs_blk* to calculate

Database and application activity monitor elements

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*.

Related reference:

- “open_loc_curs_blk - Open Local Cursors with Blocking ” on page 372
- “open_rem_curs - Open Remote Cursors ” on page 369
- “open_rem_curs_blk - Open Remote Cursors with Blocking ” on page 369
- “rej_curs_blk - Rejected Block Cursor Requests ” on page 370
- “acc_curs_blk - Accepted Block Cursor Requests ” on page 371

open_loc_curs_blk - Open Local Cursors with Blocking

Element identifier open_loc_curs_blk

Element type gauge

Table 569. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Description

The number of local blocking cursors currently open for this application.

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*.

Related reference:

- “open_loc_curs - Open Local Cursors ” on page 371
- “open_rem_curs - Open Remote Cursors ” on page 369
- “open_rem_curs_blk - Open Remote Cursors with Blocking ” on page 369
- “rej_curs_blk - Rejected Block Cursor Requests ” on page 370
- “acc_curs_blk - Accepted Block Cursor Requests ” on page 371

SQL statement activity

SQL statement activity monitor elements

The following elements provide information about SQL statement activity:

- static_sql_stmts - Static SQL Statements Attempted monitor element
- dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element

- failed_sql_stmts - Failed Statement Operations monitor element
- commit_sql_stmts - Commit Statements Attempted monitor element
- rollback_sql_stmts - Rollback Statements Attempted monitor element
- select_sql_stmts - Select SQL Statements Executed monitor element
- uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element
- ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element
- int_auto_rebinds - Internal Automatic Rebinds monitor element
- int_commits - Internal Commits monitor element
- int_rollbacks - Internal Rollbacks monitor element
- int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock monitor element
- sql_reqs_since_commit - SQL Requests Since Last Commit monitor element
- stmt_node_number - Statement Node monitor element
- binds_precompiles - Binds/Precompiles Attempted monitor element

static_sql_stmts - Static SQL Statements Attempted

Element identifier static_sql_stmts
Element type counter

Table 570. 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 571. Event Monitoring Information

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

Description

The number of static SQL statements that were attempted.

Usage You can use this element to calculate the total number of successful SQL statements at the database or application level:

$$\begin{aligned}
 & \text{dynamic_sql_stmts} \\
 & + \text{static_sql_stmts} \\
 & - \text{failed_sql_stmts} \\
 & = \text{throughput during monitoring period}
 \end{aligned}$$

Related reference:

- “failed_sql_stmts - Failed Statement Operations ” on page 374

dynamic_sql_stmts - Dynamic SQL Statements Attempted

Element identifier dynamic_sql_stmts
Element type counter

Database and application activity monitor elements

Table 572. 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 573. Event Monitoring Information

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

Description

The number of dynamic SQL statements that were attempted.

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
```

Related reference:

- “failed_sql_stmts - Failed Statement Operations ” on page 374

failed_sql_stmts - Failed Statement Operations

Element identifier failed_sql_stmts

Element type counter

Table 574. 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 575. Event Monitoring Information

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

Description

The number of SQL statements that were attempted, but failed.

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.

Related reference:

- “dynamic_sql_stmts - Dynamic SQL Statements Attempted ” on page 373
- “static_sql_stmts - Static SQL Statements Attempted ” on page 373

commit_sql_stmts - Commit Statements Attempted

Element identifier commit_sql_stmts

Element type counter

Table 576. 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 577. Event Monitoring Information

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

Description

The total number of SQL COMMIT statements that have been attempted.

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:

Database and application activity monitor elements

- 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.

Related reference:

- “int_commits - Internal Commits ” on page 380
- “rollback_sql_stmts - Rollback Statements Attempted ” on page 376
- “int_rollbacks - Internal Rollbacks ” on page 381
- “int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock ” on page 382

rollback_sql_stmts - Rollback Statements Attempted

Element identifier rollback_sql_stmts

Element type counter

Table 578. 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 579. Event Monitoring Information

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

Description

The total number of SQL ROLLBACK statements that have been attempted.

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

```

Related reference:

- “stmt_type - Statement Type ” on page 385
- “commit_sql_stmts - Commit Statements Attempted ” on page 375
- “int_commits - Internal Commits ” on page 380
- “int_rollbacks - Internal Rollbacks ” on page 381
- “int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock ” on page 382

select_sql_stmts - Select SQL Statements Executed

Element identifier select_sql_stmts

Element type counter

Table 580. 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 581. Event Monitoring Information

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

Description

The number of SQL SELECT statements that were executed.

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.

Related reference:

- “static_sql_stmts - Static SQL Statements Attempted ” on page 373
- “dynamic_sql_stmts - Dynamic SQL Statements Attempted ” on page 373

Database and application activity monitor elements

uid_sql_stmts - Update/Insert/Delete SQL Statements Executed

Element identifier uid_sql_stmts

Element type counter

Table 582. 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 583. Event Monitoring Information

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

Description

The number of SQL UPDATE, INSERT, and DELETE statements that were executed.

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.

Related reference:

- “static_sql_stmts - Static SQL Statements Attempted ” on page 373
- “dynamic_sql_stmts - Dynamic SQL Statements Attempted ” on page 373

ddl_sql_stmts - Data Definition Language (DDL) SQL Statements

Element identifier ddl_sql_stmts

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
Database	event_db	-
Connection	event_conn	-

Description

This element indicates the number of SQL Data Definition Language (DDL) statements that were executed.

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

Element identifier int_auto_rebinds

Element type counter

Table 586. 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 587. Event Monitoring Information

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

Description

The number of automatic rebinds (or recompiles) that have been attempted.

Usage Automatic rebinds are the internal binds the system performs when a package has been invalidated. The rebind is performed the first time that the database manager needs to execute an SQL statement from the package. For example, packages are invalidated when you:

- Drop an object, such as a table, view, or index, on which the plan is dependent
- Add or drop a foreign key
- Revoke object privileges on which the plan is dependent.

Database and application activity monitor elements

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.

Related reference:

- “`binds_precompiles` - Binds/Precompiles Attempted ” on page 383

int_commits - Internal Commits

Element identifier `int_commits`

Element type `counter`

Table 588. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	<code>dbase</code>	Basic
Application	<code>appl</code>	Basic

For snapshot monitoring, this counter can be reset.

Table 589. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	<code>event_db</code>	-
Connection	<code>event_conn</code>	-

Description

The total number of commits initiated internally by the database manager.

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:

$$\begin{aligned} & \text{commit_sql_stmts} \\ & + \text{int_commits} \\ & + \text{rollback_sql_stmts} \\ & + \text{int_rollbacks} \end{aligned}$$

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.

Related reference:

- “commit_sql_stmts - Commit Statements Attempted ” on page 375
- “rollback_sql_stmts - Rollback Statements Attempted ” on page 376
- “int_rollbacks - Internal Rollbacks ” on page 381

int_rollbacks - Internal Rollbacks

Element identifier int_rollbacks

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	-

Description

The total number of rollbacks initiated internally by the database manager.

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:

Database and application activity monitor elements

```
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.

Related reference:

- “commit_sql_stmts - Commit Statements Attempted ” on page 375
- “int_commits - Internal Commits ” on page 380
- “rollback_sql_stmts - Rollback Statements Attempted ” on page 376
- “int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock ” on page 382

int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock

Element identifier int_deadlock_rollbacks

Element type counter

Table 592. 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 593. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

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.

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.

Related reference:

- “deadlocks - Deadlocks Detected ” on page 303
- “rollback_sql_stmts - Rollback Statements Attempted ” on page 376
- “int_rollbacks - Internal Rollbacks ” on page 381

sql_reqs_since_commit - SQL Requests Since Last Commit

Element identifier sql_reqs_since_commit

Element type information

Table 594. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Description

Number of SQL requests that have been submitted since the last commit.

Usage You can use this element to monitor the progress of a transaction.

stmt_node_number - Statement Node

Element identifier stmt_node_number

Element type information

Table 595. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Description

Node where the statement was executed.

Usage Used to correlate each statement with the node where it was executed.

binds_precompiles - Binds/Precompiles Attempted

Element identifier binds_precompiles

Element type counter

Table 596. 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 597. Event Monitoring Information

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

Description

The number of binds and pre-compiles attempted.

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.

Related reference:

- “int_auto_rebinds - Internal Automatic Rebinds ” on page 379

xquery_stmts - XQuery Statements Attempted

Element identifier xquery_stmts

Database and application activity monitor elements

Element type counter

Table 598. 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 599. Event Monitoring Information

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

Description

The number of XQuery statements executed for an application or database.

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

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 monitor element
- stmt_operation/operation - Statement Operation monitor element
- package_name - Package Name monitor element
- package_version_id - Package Version monitor element
- consistency_token - Package Consistency Token monitor element
- section_number - Section Number monitor element
- cursor_name - Cursor Name monitor element
- creator - Application Creator monitor element
- stmt_start - Statement Operation Start Timestamp monitor element
- stmt_stop - Statement Operation Stop Timestamp monitor element
- stop_time - Event Stop Time monitor element
- start_time - Event Start Time monitor element
- stmt_elapsed_time - Most Recent Statement Elapsed Time monitor element
- stmt_text - SQL Dynamic Statement Text monitor element
- stmt_sorts - Statement Sorts monitor element
- fetch_count - Number of Successful Fetches monitor element
- sqlca - SQL Communications Area (SQLCA) monitor element
- query_card_estimate - Query Number of Rows Estimate monitor element
- query_cost_estimate - Query Cost Estimate monitor element
- stmt_history_id - Statement history identifier monitor element
stmt_history_id - Statement history identifier monitor element
- stmt_first_use_time - Statement first use time monitor
element
stmt_first_use_time - Statement first use time monitor element

Database and application activity monitor elements

- `stmt_last_use_time` – Statement last use time monitor element
`stmt_last_use_time` – Statement last use time monitor element
- `stmt_lock_timeout` - Statement lock timeout monitor element
`stmt_lock_timeout` - Statement lock timeout monitor element
- `stmt_isolation` - Statement isolation monitor element
`stmt_isolation` - Statement isolation monitor element
- `stmt_nest_level` - Statement nesting level monitor element
`stmt_nest_level` - Statement nesting level monitor element
- `stmt_invocation_id` - Statement invocation identifier monitor element
`stmt_invocation_id` - Statement invocation identifier monitor element
- `stmt_query_id` - Statement query identifier monitor element
`stmt_query_id` - Statement query identifier monitor element
- `stmt_source_id` - Statement source identifier monitor element
`stmt_source_id` - Statement source identifier monitor element
- `stmt_pkgcache_id` - Statement package cache identifier monitor element
`stmt_pkgcache_id` - Statement package cache identifier monitor element
- `comp_env_desc` - Compilation environment handle monitor element
`comp_env_desc` - Compilation environment handle monitor element
- `stmt_value_type` - Value type monitor element
`stmt_value_type` - Value type monitor element
- `stmt_value_isnull` - Value has null value monitor element
`stmt_value_isnull` - Value has null value monitor element
- `stmt_value_data` - Value data monitor element
`stmt_value_data` - Value data monitor element
- `stmt_value_index` - Value index monitor element
`stmt_value_index` - Value index monitor element

`stmt_type` - Statement Type

Element identifier	<code>stmt_type</code>
Element type	information

Table 600. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	<code>stmt</code>	Statement

Table 601. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	<code>event_detailed_dlconn</code>	-
Statements	<code>event_stmt</code>	-

Description

The type of statement processed.

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.

Database and application activity monitor elements

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.

Related reference:

- “stmt_text - SQL Dynamic Statement Text ” on page 393
- “creator - Application Creator ” on page 390
- “section_number - Section Number ” on page 389
- “package_name - Package Name ” on page 387
- “package_version_id - Package Version ” on page 388

stmt_operation/operation - Statement Operation

Element identifier stmt_operation (snapshot monitoring)
operation (event monitoring)

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	-

Description

The statement operation currently being processed or most recently processed (if none currently running).

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

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.

Related reference:

- “stmt_type - Statement Type ” on page 385
- “stmt_text - SQL Dynamic Statement Text ” on page 393
- “creator - Application Creator ” on page 390
- “section_number - Section Number ” on page 389
- “package_name - Package Name ” on page 387
- “fetch_count - Number of Successful Fetches ” on page 394
- “package_version_id - Package Version ” on page 388

package_name - Package Name

Element identifier package_name

Element type information

Table 604. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 605. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	event_detailed_dlconn	-
Statements	event_stmt	-

Description

The name of the package that contains the SQL statement currently executing.

Usage You may use this element to help identify the application program and the SQL statement that is executing.

Related reference:

- “creator - Application Creator ” on page 390
- “section_number - Section Number ” on page 389
- “stmt_text - SQL Dynamic Statement Text ” on page 393
- “package_version_id - Package Version ” on page 388

consistency_token - Package Consistency Token

Element identifier consistency_token

Element type information

Table 606. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Table 607. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-

Description

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.

Usage You can use this element to help identify the package and the SQL statement that is executing.

package_version_id - Package Version

Element identifier package_version_id

Element type information

Table 608. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Table 609. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-

Description

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).

Usage You may use this element to help identify the package and the SQL statement that is executing.

Related reference:

- "stmt_text - SQL Dynamic Statement Text " on page 393
- "creator - Application Creator " on page 390
- "package_name - Package Name " on page 387
- "section_number - Section Number " on page 389

section_number - Section Number

Element identifier section_number
 Element type information

Table 610. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcx_stmt	Statement

Table 611. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	event_detailed_dlconn	-
Statements	event_stmt	-

Description

The internal section number in the package for the SQL statement currently processing or most recently processed.

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.

Related reference:

- “stmt_text - SQL Dynamic Statement Text ” on page 393
- “creator - Application Creator ” on page 390
- “package_name - Package Name ” on page 387
- “package_version_id - Package Version ” on page 388

cursor_name - Cursor Name

Element identifier cursor_name
 Element type information

Table 612. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Database and application activity monitor elements

Table 613. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	event_detailed_dlconn	-
Statements	event_stmt	-

Description

The name of the cursor corresponding to this SQL statement.

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.

Related reference:

- “stmt_text - SQL Dynamic Statement Text ” on page 393
- “stmt_type - Statement Type ” on page 385
- “fetch_count - Number of Successful Fetches ” on page 394

creator - Application Creator

Element identifier creator

Element type information

Table 614. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 615. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_dlconn	-
Statements	event_stmt	-

Description

The authorization ID of the user that pre-compiled the application.

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.

Related reference:

- “package_name - Package Name ” on page 387
- “section_number - Section Number ” on page 389

- “package_version_id - Package Version ” on page 388

stmt_start - Statement Operation Start Timestamp

Element identifier stmt_start
 Element type timestamp

Table 616. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement, Timestamp
DCS Statement	dcs_stmt	Statement, Timestamp

Description

The date and time when the stmt_operation started executing.

Usage You can use this element with stmt_stop to calculate the elapsed statement operation execution time.

Related reference:

- “stmt_stop - Statement Operation Stop Timestamp ” on page 391
- “stmt_operation/operation - Statement Operation ” on page 386

stmt_stop - Statement Operation Stop Timestamp

Element identifier stmt_stop
 Element type Timestamp

Table 617. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement, Timestamp
DCS Statement	dcs_stmt	Statement, Timestamp

Description

The date and time when the stmt_operation stopped executing.

Usage You can use this element with stmt_start to calculate the elapsed statement operation execution time.

Related reference:

- “stmt_start - Statement Operation Start Timestamp ” on page 391
- “stmt_operation/operation - Statement Operation ” on page 386
- “stop_time - Event Stop Time ” on page 391

stop_time - Event Stop Time

Element identifier stop_time
 Element type timestamp

Table 618. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	Timestamp

Description

The date and time when the statement stopped executing.

Database and application activity monitor elements

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".

Related reference:

- "stmt_stop - Statement Operation Stop Timestamp " on page 391

start_time - Event Start Time

Element identifier start_time

Element type timestamp

Table 619. 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

Description

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.

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".

Related reference:

- "stmt_operation/operation - Statement Operation " on page 386

stmt_elapsed_time - Most Recent Statement Elapsed Time

Element identifier stmt_elapsed_time

Element type time

Table 620. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement, Timestamp
DCS Statement	dcs_stmt	Statement, Timestamp

Description

The elapsed execution time of the most recently completed statement.

Usage Use this element as an indicator of the time it takes for a statement to complete.

Related reference:

- “gw_comm_errors - Communication Errors ” on page 471
- “gw_comm_error_time - Communication Error Time ” on page 471

stmt_text - SQL Dynamic Statement Text

Element identifier stmt_text
Element type information

Table 621. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
Dynamic SQL	dynsql	Basic
DCS Statement	dcs_stmt	Statement

Table 622. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	event_detailed_dlconn	-
Statements	event_stmt	-

Description

This is the text of the dynamic SQL statement.

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 event monitors, this element is returned only for dynamic statements. 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.

See *section_number* for information on how to query the system catalog tables to obtain static SQL statement text that is not provided due to performance considerations.

Related reference:

- “stmt_operation/operation - Statement Operation ” on page 386
- “cursor_name - Cursor Name ” on page 389
- “input_db_alias - Input Database Alias ” on page 420
- “creator - Application Creator ” on page 390
- “package_name - Package Name ” on page 387
- “section_number - Section Number ” on page 389
- “package_version_id - Package Version ” on page 388

stmt_sorts - Statement Sorts

Element identifier stmt_sorts

Element type counter

Table 623. Snapshot Monitoring Information

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

Description

The total number of times that a set of data was sorted in order to process the stmt_operation.

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.

Related reference:

- “total_sorts - Total Sorts ” on page 210

fetch_count - Number of Successful Fetches

Element identifier fetch_count

Element type counter

Table 624. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 625. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-

Description

For the stmt snapshot monitoring level 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). That is, fetch_count represents the number of times the server needed to send a reply data back to the gateway while processing a statement.

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 generated a statement event record for every FETCH statement. A record event is only generated when a FETCH returns a non-zero SQLCODE.

Related reference:

- "stmt_type - Statement Type " on page 385
- "stmt_operation/operation - Statement Operation " on page 386
- "cursor_name - Cursor Name " on page 389
- "stmt_start - Statement Operation Start Timestamp " on page 391
- "stmt_stop - Statement Operation Stop Timestamp " on page 391

sqlca - SQL Communications Area (SQLCA)

Element identifier sqlca
Element type information

Table 626. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-

Description

The SQLCA data structure that was returned to the application at statement completion.

Usage The SQLCA data structure can be used to determined if the statement completed successfully. See the *SQL Reference* or *Administrative API Reference* for information about the content of the SQLCA.

Related reference:

- "stmt_operation/operation - Statement Operation " on page 386

query_card_estimate - Query Number of Rows Estimate

Element identifier query_card_estimate
Element type information

Database and application activity monitor elements

Table 627. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Description

An estimate of the number of rows that will be returned by a query.

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.

Related reference:

- “query_cost_estimate - Query Cost Estimate ” on page 396

query_cost_estimate - Query Cost Estimate

Element identifier query_cost_estimate

Element type information

Table 628. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Description

Estimated cost, in timerons, for a query, as determined by the SQL compiler.

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.

Database and application activity monitor elements

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

Element identifier stmt_history_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 Values	event_data_value	-
Deadlocks with Details History	event_stmt_history	-

Description

This numerical element shows the position in which the statement was run within the current unit of work, 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.

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

Element identifier stmt_first_use_time

Element type information

Table 630. 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

Description

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.

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

Element identifier stmt_last_use_time

Element type information

Table 631. 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

Description

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.

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

Element identifier stmt_lock_timeout

Element type information

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	-

Description

This element shows the lock timeout value in effect for the statement while it was being run.

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

Element identifier stmt_isolation

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	-

Description

This element shows the isolation value in effect for the statement while it was being run.

The list of 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

Element identifier stmt_nest_level

Element type information

Table 634. 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	-

Description

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).

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

Element identifier stmt_invocation_id

Element type information

Table 635. 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	-

Description

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.

Usage You can use this element, along with stmt_nest_level, to uniquely identify

Database and application activity monitor elements

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

Element identifier stmt_query_id
Element type information

Table 636. 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	-

Description

This element shows the internal query identifier (ID) given to any SQL statement used as a cursor.

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

Element identifier stmt_source_id
Element type information

Table 637. 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	-

Description

This element shows the internal identifier (ID) given to the source of the SQL statement that was run.

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

Element identifier stmt_pkgcache_id
Element type information

Table 638. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details History Values	event_stmt_history	-

Table 638. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details History	event_stmt_history	-

Description

This element shows the internal package cache identifier (ID) for a dynamic SQL statement.

Usage You can use this element to uniquely identify a particular SQL statement. You can also use this element in conjunction with other statement history entries to understand the cause of the deadlock.

comp_env_desc - Compilation environment handle

Element identifier comp_env_desc

Element type information

Table 639. 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	-

Description

This element represents a handle to the compilation environment used when compiling the SQL statement.

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

Element identifier stmt_value_type

Element type information

Table 640. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details History Values	stmt_value_type	-

Description

This element contains a string representation of the type of a data value associated with an SQL statement.

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

Element identifier stmt_value_isnull

Element type information

Database and application activity monitor elements

Table 641. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details History Values	stmt_value_isnull	-

Description

This element shows whether a data value associated with an SQL statement is the NULL value.

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

Element identifier stmt_value_data

Element type information

Table 642. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details History Values	stmt_value_data	-

Description

This element contains a string representation of a data value to an SQL statement. LOB, LONG, DATALINK, and structured type parameters appear as empty strings. Date, time, and timestamp fields are recorded in ISO format.

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

Element identifier stmt_value_index

Element type information

Table 643. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details History Values	stmt_value_data	-

Description

This element represents the position of the input parameter marker or host variable used in the SQL statement.

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

Element identifier inact_stmthist_sz

Element type information

Table 644. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
application	appl	-
database	db	-

Description

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.

Usage You can use this element when tuning the database monitor heap.

stmt_value_isreoptvalue - Variable used for statement reoptimization

Element identifier stmt_value_isreoptvalue

Element type information

Table 645. Snapshot Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details History Values	stmt_value_isreoptvalue	-
database	db	-

Description

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.

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

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 monitor element
- ss_node_number - Subsection Node Number monitor element
- ss_status - Subsection Status monitor element
- ss_exec_time - Subsection Execution Elapsed Time monitor element

Database and application activity monitor elements

- tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue monitor element
- tq_node_waited_for - Waited for Node on a Tablequeue monitor element
- tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed monitor element
- tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed monitor element
- tq_rows_read - Number of Rows Read from Tablequeues monitor element
- tq_rows_written - Number of Rows Written to Tablequeues monitor element
- tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows monitor element
- tq_id_waiting_on - Waited on Node on a Tablequeue monitor element

ss_number - Subsection Number

Element identifier ss_number
Element type information

Table 646. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 647. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

Description

Identifies the subsection associated with the returned information.

Usage This number relates to the subsection number in the access plan that can be obtained with db2expln.

ss_node_number - Subsection Node Number

Element identifier ss_node_number
Element type information

Table 648. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 649. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

Description

Node where the subsection was executed.

Usage Use to correlate each subsection with the database partition where it was executed.

ss_status - Subsection Status

Element identifier ss_status

Element type information

Table 650. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Description

The current status of an executing subsection.

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

Related reference:

- “tq_node_waited_for - Waited for Node on a Tablequeue ” on page 406
- “tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue ” on page 405

ss_exec_time - Subsection Execution Elapsed Time

Element identifier ss_exec_time

Element type counter

Table 651. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 652. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

Description

The time in seconds that it took a subsection to execute.

Usage Allows you to track the progress of a subsection.

tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue

Element identifier tq_wait_for_any

Element type information

Table 653. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Description

This flag is used to indicate that the subsection is blocked because it is waiting to receive rows from any node.

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

Database and application activity monitor elements

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.

Related reference:

- “ss_status - Subsection Status ” on page 404
- “tq_node_waited_for - Waited for Node on a Tablequeue ” on page 406

tq_node_waited_for - Waited for Node on a Tablequeue

Element identifier tq_node_waited_for

Element type information

Table 654. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Description

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.

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.

Related reference:

- “ss_status - Subsection Status ” on page 404
- “tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue ” on page 405

tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed

Element identifier tq_tot_send_spills

Element type counter

Table 655. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 656. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

Description

Total number of tablequeue buffers overflowed to a temporary table.

Usage Indicates the total number of tablequeue buffers that have been written to a temporary table. See tq_cur_send_spills for more information.

Related reference:

- “ss_status - Subsection Status ” on page 404
- “tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed ” on page 407

tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed

Element identifier tq_cur_send_spills

Element type gauge

Table 657. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Description

Current number of tablequeue buffers residing in a temporary table.

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.

Related reference:

- “ss_status - Subsection Status ” on page 404
- “tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed ” on page 406

tq_rows_read - Number of Rows Read from Tablequeues

Element identifier tq_rows_read

Element type counter

Table 658. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 659. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

Database and application activity monitor elements

Description

Total number of rows read from tablequeues.

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

Element identifier tq_rows_written

Element type counter

Table 660. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 661. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

Description

Total number of rows written to tablequeues.

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

Element identifier tq_max_send_spills

Element type water mark

Table 662. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 663. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

Description

Maximum number of tablequeue buffers overflowed to a temporary table.

Usage Indicates the maximum number of tablequeue buffers that have been written to a temporary table.

Related reference:

- “tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed ” on page 406
- “tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed ” on page 407

tq_id_waiting_on - Waited on Node on a Tablequeue

Element identifier tq_id_waiting_on

Element type information

Table 664. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Description

The agent that is waiting.

Usage This can be used for troubleshooting.

Related reference:

- “ss_status - Subsection Status ” on page 404
- “tq_node_waited_for - Waited for Node on a Tablequeue ” on page 406

Dynamic SQL

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 monitor element
- num_compilations - Statement Compilations monitor element
- prep_time_worst - Statement Worst Preparation Time monitor element
- prep_time_best - Statement Best Preparation Time monitor element
- total_exec_time - Elapsed Statement Execution Time monitor element

num_executions - Statement Executions

Element identifier num_executions

Element type counter

Table 665. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Basic

For snapshot monitoring, this counter can be reset.

Description

The number of times that an SQL statement has been executed.

Usage You can use this element to identify the most frequently executed SQL statements in your system.

Database and application activity monitor elements

Related reference:

- “num_compilations - Statement Compilations ” on page 410

num_compilations - Statement Compilations

Element identifier num_compilations

Element type counter

Table 666. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Basic

Description

The number of different compilations for a specific SQL statement.

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.

Related reference:

- “num_executions - Statement Executions ” on page 409

prep_time_worst - Statement Worst Preparation Time

Element identifier prep_time_worst

Element type water mark

Table 667. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Basic

Description

The longest amount of time in microseconds that was required to prepare a specific SQL statement.

Usage Use this value in conjunction with prep_time_best to identify SQL statements that are expensive to compile.

Related reference:

- “prep_time_best - Statement Best Preparation Time ” on page 410

prep_time_best - Statement Best Preparation Time

Element identifier prep_time_best

Element type water mark

Table 668. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Basic

Description

The shortest amount of time that was required to prepare a specific SQL statement.

Usage Use this value in conjunction with `prep_time_worst` to identify SQL statements that are expensive to compile.

Related reference:

- “`prep_time_worst` - Statement Worst Preparation Time ” on page 410

total_exec_time - Elapsed Statement Execution Time

Element identifier total_exec_time
Element type time

Table 669. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Statement

For snapshot monitoring, this counter can be reset.

Description

The total time in seconds and microseconds that was spent executing a particular statement in the SQL cache.

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.

Related reference:

- “`num_executions` - Statement Executions ” on page 409
- “`num_compilations` - Statement Compilations ” on page 410
- “`total_sys_cpu_time` - Total System CPU for a Statement ” on page 418
- “`total_usr_cpu_time` - Total User CPU for a Statement ” on page 419

Intra-query parallelism

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 monitor element
- `agents_top` - Number of Agents Created monitor element
- `degree_parallelism` - Degree of Parallelism monitor element

num_agents - Number of Agents Working on a Statement

Element identifier num_agents
Element type gauge

Table 670. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
Application	subsection	Statement

Description

Number of concurrent agents currently executing a statement or subsection.

Database and application activity monitor elements

Usage An indicator how well the query is parallelized. This is useful for tracking the progress of query execution, by taking successive snapshots.

Related reference:

- “agents_top - Number of Agents Created ” on page 412
- “degree_parallelism - Degree of Parallelism ” on page 412

agents_top - Number of Agents Created

Element identifier agents_top

Element type water mark

Table 671. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Statement
Application	stmt	Statement

Description

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.

Usage An indicator how well intra-query parallelism was realized.

Related reference:

- “num_agents - Number of Agents Working on a Statement ” on page 411
- “degree_parallelism - Degree of Parallelism ” on page 412

degree_parallelism - Degree of Parallelism

Element identifier degree_parallelism

Element type information

Table 672. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Description

The degree of parallelism requested when the query was bound.

Usage Use with agents_top, to determine if the query achieved maximum level of parallelism.

Related reference:

- “num_agents - Number of Agents Working on a Statement ” on page 411
- “agents_top - Number of Agents Created ” on page 412

CPU usage

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 monitor element
- agent_sys_cpu_time - System CPU Time used by Agent monitor element
- stmt_usr_cpu_time - User CPU Time used by Statement monitor element
- stmt_sys_cpu_time - System CPU Time used by Statement monitor element
- user_cpu_time - User CPU Time monitor element
- system_cpu_time - System CPU Time monitor element
- ss_usr_cpu_time - User CPU Time used by Subsection monitor element
- ss_sys_cpu_time - System CPU Time used by Subsection monitor element
- total_sys_cpu_time - Total System CPU for a Statement monitor element
- total_usr_cpu_time - Total User CPU for a Statement monitor element

agent_usr_cpu_time - User CPU Time used by Agent

Element identifier agent_usr_cpu_time
Element type time

Table 673. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Timestamp

For snapshot monitoring, this counter can be reset.

Description

The total CPU time (in seconds and microseconds) used by database manager agent process.

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.

Related reference:

- “agent_sys_cpu_time - System CPU Time used by Agent ” on page 413
- “stmt_usr_cpu_time - User CPU Time used by Statement ” on page 414
- “stmt_sys_cpu_time - System CPU Time used by Statement ” on page 415
- “ss_usr_cpu_time - User CPU Time used by Subsection ” on page 417
- “ss_sys_cpu_time - System CPU Time used by Subsection ” on page 418
- “user_cpu_time - User CPU Time ” on page 416
- “system_cpu_time - System CPU Time ” on page 416
- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

agent_sys_cpu_time - System CPU Time used by Agent

Element identifier agent_sys_cpu_time
Element type time

Database and application activity monitor elements

Table 674. 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.

Description

The total *system* CPU time (in seconds and microseconds) used by the database manager agent process.

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.

Related reference:

- “agent_usr_cpu_time - User CPU Time used by Agent ” on page 413
- “stmt_usr_cpu_time - User CPU Time used by Statement ” on page 414
- “stmt_sys_cpu_time - System CPU Time used by Statement ” on page 415
- “ss_usr_cpu_time - User CPU Time used by Subsection ” on page 417
- “ss_sys_cpu_time - System CPU Time used by Subsection ” on page 418
- “user_cpu_time - User CPU Time ” on page 416
- “system_cpu_time - System CPU Time ” on page 416
- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

stmt_usr_cpu_time - User CPU Time used by Statement

Element identifier stmt_usr_cpu_time

Element type time

Table 675. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement, Timestamp
Application	stmt	Statement, Timestamp

Description

The total *user* CPU time (in seconds and microseconds) used by the currently executing statement.

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.

Database and application activity monitor elements

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.

Related reference:

- “agent_sys_cpu_time - System CPU Time used by Agent ” on page 413
- “agent_usr_cpu_time - User CPU Time used by Agent ” on page 413
- “stmt_sys_cpu_time - System CPU Time used by Statement ” on page 415
- “ss_usr_cpu_time - User CPU Time used by Subsection ” on page 417
- “ss_sys_cpu_time - System CPU Time used by Subsection ” on page 418
- “user_cpu_time - User CPU Time ” on page 416
- “system_cpu_time - System CPU Time ” on page 416
- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

stmt_sys_cpu_time - System CPU Time used by Statement

Element identifier stmt_sys_cpu_time

Element type time

Table 676. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement, Timestamp
Application	stmt	Statement, Timestamp

Description

The total *system* CPU time (in seconds and microseconds) used by the currently executing statement.

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.

Related reference:

- “agent_sys_cpu_time - System CPU Time used by Agent ” on page 413
- “stmt_usr_cpu_time - User CPU Time used by Statement ” on page 414
- “agent_usr_cpu_time - User CPU Time used by Agent ” on page 413
- “ss_usr_cpu_time - User CPU Time used by Subsection ” on page 417
- “ss_sys_cpu_time - System CPU Time used by Subsection ” on page 418
- “user_cpu_time - User CPU Time ” on page 416
- “system_cpu_time - System CPU Time ” on page 416

Database and application activity monitor elements

- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

user_cpu_time - User CPU Time

Element identifier user_cpu_time

Element type time

Table 677. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-
Transactions	event_xact	-
Statements	event_stmt	-

Description

The total *user* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement.

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.

Related reference:

- “agent_sys_cpu_time - System CPU Time used by Agent ” on page 413
- “stmt_usr_cpu_time - User CPU Time used by Statement ” on page 414
- “stmt_sys_cpu_time - System CPU Time used by Statement ” on page 415
- “ss_usr_cpu_time - User CPU Time used by Subsection ” on page 417
- “ss_sys_cpu_time - System CPU Time used by Subsection ” on page 418
- “agent_usr_cpu_time - User CPU Time used by Agent ” on page 413
- “system_cpu_time - System CPU Time ” on page 416
- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

system_cpu_time - System CPU Time

Element identifier system_cpu_time

Element type time

Table 678. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-
Transactions	event_xact	-
Statements	event_stmt	-

Description

The total *system* CPU time (in seconds and microseconds) used by the database manager agent process, the unit of work, or the statement.

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.

Related reference:

- “agent_sys_cpu_time - System CPU Time used by Agent ” on page 413
- “stmt_usr_cpu_time - User CPU Time used by Statement ” on page 414
- “stmt_sys_cpu_time - System CPU Time used by Statement ” on page 415
- “ss_usr_cpu_time - User CPU Time used by Subsection ” on page 417
- “ss_sys_cpu_time - System CPU Time used by Subsection ” on page 418
- “user_cpu_time - User CPU Time ” on page 416
- “agent_usr_cpu_time - User CPU Time used by Agent ” on page 413
- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

ss_usr_cpu_time - User CPU Time used by Subsection

Element identifier ss_usr_cpu_time

Element type time

Table 679. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Timestamp

Table 680. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	Timestamp

Description

The total user CPU time (in seconds and microseconds) used by the currently executing statement subsection.

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.

Related reference:

- “agent_sys_cpu_time - System CPU Time used by Agent ” on page 413
- “stmt_usr_cpu_time - User CPU Time used by Statement ” on page 414
- “stmt_sys_cpu_time - System CPU Time used by Statement ” on page 415
- “agent_usr_cpu_time - User CPU Time used by Agent ” on page 413
- “ss_sys_cpu_time - System CPU Time used by Subsection ” on page 418
- “user_cpu_time - User CPU Time ” on page 416
- “system_cpu_time - System CPU Time ” on page 416
- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

ss_sys_cpu_time - System CPU Time used by Subsection

Element identifier ss_sys_cpu_time

Element type time

Table 681. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Timestamp

Table 682. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	Timestamp

Description

The total system CPU time (in seconds and microseconds) used by the currently executing statement subsection.

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.

Related reference:

- “agent_sys_cpu_time - System CPU Time used by Agent ” on page 413
- “stmt_usr_cpu_time - User CPU Time used by Statement ” on page 414
- “stmt_sys_cpu_time - System CPU Time used by Statement ” on page 415
- “ss_usr_cpu_time - User CPU Time used by Subsection ” on page 417
- “user_cpu_time - User CPU Time ” on page 416
- “agent_usr_cpu_time - User CPU Time used by Agent ” on page 413
- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419

total_sys_cpu_time - Total System CPU for a Statement

Element identifier total_sys_cpu_time

Element type time

Table 683. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Statement

For snapshot monitoring, this counter can be reset.

Description

The total system CPU time for an SQL statement.

Usage Use this element with Elapsed Statement Execution Time and Total User CPU for a Statement to evaluate which statements are the most expensive.

Related reference:

- “total_usr_cpu_time - Total User CPU for a Statement ” on page 419
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

total_usr_cpu_time - Total User CPU for a Statement

Element identifier total_usr_cpu_time

Element type time

Table 684. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Dynamic SQL	dynsql	Statement

For snapshot monitoring, this counter can be reset.

Description

The total user CPU time for an SQL statement.

Usage Use this element with Elapsed Statement Execution Time and to evaluate the longest running statements.

Related reference:

- “total_sys_cpu_time - Total System CPU for a Statement ” on page 418
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

Snapshot monitoring

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 monitor element
- input_db_alias - Input Database Alias monitor element
- time_stamp - Snapshot Time monitor element
- num_nodes_in_db2_instance - Number of Nodes in Partition monitor element

last_reset - Last Reset Timestamp

Element identifier last_reset

Element type timestamp

Table 685. 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

Description

Indicates the date and time that the monitor counters were reset for the application issuing the GET SNAPSHOT.

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.

Database and application activity monitor elements

The database manager counters will only be reset if you reset all active databases.

Related reference:

- “input_db_alias - Input Database Alias ” on page 420

input_db_alias - Input Database Alias

Element identifier input_db_alias

Element type information

Table 686. 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

Description

The alias of the database provided when calling the snapshot function.

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.

Related reference:

- “last_reset - Last Reset Timestamp ” on page 419
- “client_db_alias - Database Alias Used by Application ” on page 174

time_stamp - Snapshot Time

Element identifier time_stamp

Element type timestamp

Table 687. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Description

The date and time when the database system monitor information was collected.

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

Element identifier num_nodes_in_db2_instance

Element type information

Table 688. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Table 689. Event Monitoring Information

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

Description

The number of nodes on the instance where the snapshot was taken.

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

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 monitor element
- first_overflow_time - Time of First Event Overflow monitor element
- last_overflow_time - Time of Last Event Overflow monitor element
- byte_order - Byte Order of Event Data monitor element
- version - Version of Monitor Data monitor element
- event_monitor_name - Event Monitor Name monitor element
- partial_record - Partial Record monitor element
- event_time - Event Time monitor element
- evmon_flushes - Number of Event Monitor Flushes monitor element
- evmon_activates - Number of Event Monitor Activations monitor element
- sql_req_id - Request Identifier for SQL Statement monitor element
- message - Control Table Message monitor element
- message_time - Timestamp Control Table Message monitor element
- partition_number - Partition Number monitor element

count - Number of Event Monitor Overflows

Element identifier count

Element type counter

Table 690. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Overflow Record	event_overflow	-

Description

The number of consecutive overflows that have occurred.

Usage You may use this element to get an indication of how much monitor data has been lost.

Database and application activity monitor elements

The event monitor sends one overflow record for a set of consecutive overflows.

first_overflow_time - Time of First Event Overflow

Element identifier first_overflow_time

Element type timestamp

Table 691. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Overflow Record	event_overflow	-

Description

The date and time of the first overflow recorded by this overflow record.

Usage Use this element with *last_overflow_time* to calculate the elapsed time for which the overflow record was generated.

Related reference:

- “count - Number of Event Monitor Overflows ” on page 421

last_overflow_time - Time of Last Event Overflow

Element identifier last_overflow_time

Element type timestamp

Table 692. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Overflow Record	event_overflow	-

Description

The date and time of the last overflow recorded this overflow record.

Usage Use this element with *first_overflow_time* to calculate the elapsed time for which the overflow record was generated.

Related reference:

- “count - Number of Event Monitor Overflows ” on page 421

byte_order - Byte Order of Event Data

Element identifier byte_order

Element type information

Table 693. Event Monitoring Information

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

Description

The byte ordering of numeric data, specifically whether the event data stream was generated on a “big endian” server (for example, a RISC System/6000®) or “little endian” server (for example, an Intel®-based PC running Windows 2000).

Usage This information is needed to allow you to interpret numeric data in the

Database and application activity monitor elements

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

Element identifier version
Element type information

Table 694. Event Monitoring Information

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

Description

The version of the database manager that produced the event monitor data stream.

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_VERSION8.

event_monitor_name - Event Monitor Name

Element identifier event_monitor_name
Element type information

Table 695. Event Monitoring Information

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

Description

The name of the event monitor that created the event data stream.

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

Element identifier partial_record
Element type information

Database and application activity monitor elements

Table 696. 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	-

Description

Indicates that an event monitor record is only a partial record.

Usage Most event monitors do not output their results until database deactivation. You can use the FLUSH EVENT MONITORS 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.

event_time - Event Time

Element identifier event_time

Element type information

Table 697. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tablespaces	event_tablespace	-
Tables	event_table	-

Description

The date and time an event occurred.

Usage You can use this element to help relate events chronologically.

evmon_flushes - Number of Event Monitor Flushes

Element identifier evmon_flushes

Element type information

Table 698. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
Tables	event_table	-
Tablespaces	event_tablespace	-
Bufferpools	event_bufferpool	-

Description

The number of times the FLUSH EVENT MONITOR SQL statement has been issued.

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

Element identifier evmon_activates

Element type counter

Table 699. 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	-

Description

The number of times an event monitor has been activated.

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

Element identifier sql_req_id

Element type information

Table 700. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-

Description

The request identifier for an operation in an SQL statement.

Database and application activity monitor elements

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

Element identifier message
Element type information

Table 701. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
-	-	-

Description

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.

Usage The following are possible values:

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.

message_time - Timestamp Control Table Message

Element identifier message_time
Element type timestamp

Table 702. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
-	-	-

Description

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.

partition_number - Partition Number

Element identifier partition_number
Element type information

Table 703. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
-	-	-

Description

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.

High availability disaster recovery

High availability disaster recovery 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 monitor element
- `hadr_state` - HADR State monitor element
- `hadr_syncmode` - HADR Synchronization Mode monitor element
- `hadr_connect_status` - HADR Connection Status monitor element
- `hadr_connect_time` - HADR Connection Time monitor element
- `hadr_heartbeat` - HADR Heartbeat monitor element
- `hadr_local_host` - HADR Local Host monitor element
- `hadr_local_service` - HADR Local Service monitor element
- `hadr_remote_host` - HADR Remote Host monitor element
- `hadr_remote_service` - HADR Remote Service monitor element
- `hadr_remote_instance` - HADR Remote Instance monitor element
- `hadr_timeout` - HADR Timeout monitor element
- `hadr_primary_log_file` - HADR Primary Log File monitor element
- `hadr_primary_log_page` - HADR Primary Log Page monitor element
- `hadr_primary_log_lsn` - HADR Primary Log LSN monitor element
- `hadr_standby_log_file` - HADR Standby Log File monitor element
- `hadr_standby_log_page` - HADR Standby Log Page monitor element
- `hadr_standby_log_lsn` - HADR Standby Log LSN monitor element
- `hadr_log_gap` - HADR Log Gap monitor element

Related concepts:

- “High availability disaster recovery overview” in *Data Recovery and High Availability Guide and Reference*

`hadr_role` - HADR Role

Element identifier	<code>hadr_role</code>
Element type	information

Table 704. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	<code>hadr</code>	Basic

High availability disaster recovery monitor elements

Description

The current HADR role of the 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.

Usage Use this element to determine the HADR role of a database.

hadr_state - HADR State monitor element

Element identifier `hadr_state`

Element type `information`

Table 705. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The current HADR state of the database. The data type of this element is integer. This element should be ignored if the database's HADR role is standard. 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 catch-up.
- `SQLM_HADR_STATE_REM_CATCH_PEND`: the database is waiting to connect to its partner to do remote catch-up.
- `SQLM_HADR_STATE_REM_CATCHUP`: the database is doing remote catch-up.
- `SQLM_HADR_STATE_PEER`: the primary and standby databases are connected and are in peer state.

Usage Use this element to determine the HADR state of a database. Use the `hadr_role` monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_syncmode - HADR Synchronization Mode monitor element

Element identifier `hadr_syncmode`

Element type `information`

Table 706. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The current HADR synchronization mode of the database. The data type of this element is integer. This element should be ignored if the database's HADR role is standard. 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.

Usage Use this element to determine the HADR synchronization mode of a database. Use the *hadr_role* monitor element to determine the HADR role of the database.

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.

Related reference:

- “hadr_role - HADR Role ” on page 427

hadr_connect_status - HADR Connection Status monitor element

Element identifier	hadr_connect_status
Element type	information

Table 707. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The current HADR connection status of the database. The data type of this element is integer. This element should be ignored if the database's HADR role is standard. 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.

High availability disaster recovery monitor elements

Usage Use this element to determine the HADR connection status of a database. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “hadr_role - HADR Role ” on page 427

hadr_connect_time - HADR Connection Time monitor element

Element identifier hadr_connect_time
Element type timestamp

Table 708. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

Shows one of the following:

- HADR connection time
- HADR congestion time
- HADR disconnection time

This element should be ignored if the database’s HADR role is standard. 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.

Usage Use this element to determine when the current HADR connection status began. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “hadr_connect_status - HADR Connection Status monitor element” on page 429
- “hadr_role - HADR Role ” on page 427
- “dft_monswitches - Default database system monitor switches configuration parameter” in *Performance Guide*

hadr_heartbeat - HADR Heartbeat monitor element

Element identifier hadr_heartbeat

Element type counter

Table 709. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

For snapshot monitoring, this counter cannot be reset.

Description

Number of missed heartbeats on the HADR connection. If the database is in HADR primary or standby role, this element indicates the health of the HADR connection. A heartbeat is a message sent from the other HADR database at regular intervals. If the value for this element is zero, no heartbeats have been missed and the connection is healthy. The higher the value, the worse the condition of the connection.

An HADR database expects at least one heartbeat message from the other database in each quarter of the time interval defined in the HADR_TIMEOUT database configuration parameter, or in 30 seconds, whichever is shorter. For example, if the HADR_TIMEOUT value is 80 (seconds), then the HADR database expects at least one heartbeat message from the other database every 20 seconds.

Notes:

1. The data type of this element is integer.
2. This element should be ignored if the database's HADR role is standard.

Usage Use this element to determine the health of the HADR connection. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_local_host - HADR Local Host monitor element

Element identifier hadr_local_host

Element type information

Table 710. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The local HADR host name. The value is displayed as a host name string or an IP address string such as "1.2.3.4". This element should be ignored if the database's HADR role is standard.

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.

Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “*hadr_role* - HADR Role ” on page 427

hadr_local_service - HADR Local Service monitor element

Element identifier	hadr_local_service
Element type	information

Table 711. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The local HADR TCP service. This value is displayed as a service name string or a port number string. This element should be ignored if the database’s HADR role is standard.

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.

Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “*hadr_role* - HADR Role ” on page 427

hadr_remote_host - HADR Remote Host monitor element

Element identifier	hadr_remote_host
Element type	information

Table 712. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The remote HADR host name. The value is displayed as a host name string or an IP address string such as “1.2.3.4”. This element should be ignored if the database’s HADR role is standard.

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.

Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “*hadr_role* - HADR Role ” on page 427

hadr_remote_service - HADR Remote Service monitor element

Element identifier	hadr_remote_service
Element type	information

Table 713. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The remote HADR TCP service. This value is displayed as a service name string or a port number string. This element should be ignored if the database's HADR role is standard.

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.

Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_remote_instance - HADR Remote Instance monitor element

Element identifier	hadr_remote_instance
Element type	information

Table 714. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The remote HADR instance name. This element should be ignored if the database's HADR role is standard.

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.

Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_timeout - HADR Timeout monitor element

Element identifier	hadr_timeout
--------------------	--------------

High availability disaster recovery monitor elements

Element type information

Table 715. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The number of seconds it takes for an HADR database server to consider a communication attempt has failed. For an attempt to fail, an HADR database server must not receive a reply message from its partner within the number of seconds listed by this element. This element should be ignored if the database's HADR role is standard.

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.

Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_primary_log_file - HADR Primary Log File monitor element

Element identifier *hadr_primary_log_file*

Element type information

Table 716. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The name of the current log file on the primary HADR database. This element should be ignored if the database's HADR role is standard.

Usage Use this element to determine the current log file on the primary HADR database. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_primary_log_page - HADR Primary Log Page monitor element

Element identifier *hadr_primary_log_page*

Element type information

Table 717. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

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. This element should be ignored if the database's HADR role is standard.

Usage Use this element to determine the current log page on the primary HADR database. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_primary_log_lsn - HADR Primary Log LSN monitor element

Element identifier	hadr_primary_log_lsn
Element type	information

Table 718. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The current log position of the primary HADR database. Log sequence number (LSN) is a byte offset in the database's log stream. This element should be ignored if the database's HADR role is standard.

Usage Use this element to determine the current log position on the primary HADR database. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

hadr_standby_log_file - HADR Standby Log File monitor element

Element identifier	hadr_standby_log_file
Element type	information

Table 719. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The name of the current log file on the standby HADR database. This element should be ignored if the database's HADR role is standard.

High availability disaster recovery monitor elements

Usage Use this element to determine the current log file on the standby HADR database. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “hadr_role - HADR Role ” on page 427

hadr_standby_log_page - HADR Standby Log Page monitor element

Element identifier hadr_standby_log_page

Element type information

Table 720. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

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. This element should be ignored if the database’s HADR role is standard.

Usage Use this element to determine the current log page on the standby HADR database. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “hadr_role - HADR Role ” on page 427

hadr_standby_log_lsn - HADR Standby Log LSN monitor element

Element identifier hadr_standby_log_lsn

Element type information

Table 721. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

The current log position of the standby HADR database. Log sequence number (LSN) is a byte offset in the database’s log stream. This element should be ignored if the database’s HADR role is standard.

Usage Use this element to determine the current log position on the standby HADR database. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- “hadr_role - HADR Role ” on page 427

hadr_log_gap - HADR Log Gap

Element identifier	hadr_log_gap
Element type	information

Table 722. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	hadr	Basic

Description

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.

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.

This element should be ignored if the database's HADR role is standard.

Usage Use this element to determine the gap between the primary and standby HADR database logs. Use the *hadr_role* monitor element to determine the HADR role of the database.

Related reference:

- "hadr_role - HADR Role " on page 427

DB2 Connect

DB2 Connect monitor elements

The following elements provide DB2 Connection information at the database, application, transaction, and statement levels:

- dcs_db_name - DCS Database Name monitor element
- host_db_name - Host Database Name monitor element
- gw_db_alias - Database Alias at the Gateway monitor element
- gw_con_time - DB2 Connect Gateway First Connect Initiated monitor element
- gw_connections_top - Maximum Number of Concurrent Connections to Host Database monitor element
- gw_total_cons - Total Number of Attempted Connections for DB2 Connect monitor element
- gw_cur_cons - Current Number of Connections for DB2 Connect monitor element
- gw_cons_wait_host - Number of Connections Waiting for the Host to Reply monitor element
- gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request monitor element
- gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing monitor element
- sql_stmts - Number of SQL Statements Attempted monitor element

DB2 Connect monitor elements

- sql_chains - Number of SQL Chains Attempted monitor elementsql_chains - Number of SQL Chains Attempted monitor element
- open_cursors - Number of Open Cursors monitor element
- dcs_appl_status - DCS Application Status monitor element
- agent_status - DCS Application Agents monitor element
- host_ccsid - Host Coded Character Set ID monitor element
- outbound_comm_protocol - Outbound Communication Protocol monitor element
- outbound_comm_address - Outbound Communication Address monitor element
- inbound_comm_address - Inbound Communication Address monitor element
- inbound_bytes_received - Inbound Number of Bytes Received monitor element
- outbound_bytes_sent - Outbound Number of Bytes Sent monitor element
- outbound_bytes_received - Outbound Number of Bytes Received monitor element
- inbound_bytes_sent - Inbound Number of Bytes Sent monitor element
- outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent monitor element
- outbound_bytes_received_top - Maximum Outbound Number of Bytes Received monitor element
- outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent monitor element
- outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received monitor element
- max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes monitor element
- max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes monitor element
- max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes monitor element
- max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes monitor element
- max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes monitor element
- max_data_received_512 - Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes monitor element
- max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes monitor element
- max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes monitor element
- max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes monitor element
- max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes monitor element
- max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes monitor element
- max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes monitor element
- max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes monitor element

- max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes monitor element
- max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes monitor element
- max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes monitor element
- max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes monitor element
- max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes monitor element
- max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes monitor element
- max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes monitor element
- max_data_sent_gt64000 - Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes monitor element
- max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes monitor element
- max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms monitor element
- max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms monitor element
- max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms monitor element
- max_network_time_16_ms - Number of Statements with Network Time between 4 and 16 ms monitor element
- max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms monitor element
- max_network_time_100_ms - Number of Statements with Network Time between 16 and 100 ms monitor element
- max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms monitor element
- max_network_time_500_ms - Number of Statements with Network Time between 100 and 500 ms monitor element
- max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms monitor element
- max_network_time_gt500_ms - Number of Statements with Network Time greater than 500 ms monitor element
- network_time_top - Maximum Network Time for Statement monitor element
- network_time_bottom - Minimum Network Time for Statement monitor element
- xid - Transaction ID monitor element
- elapsed_exec_time - Statement Execution Elapsed Time monitor element
- host_response_time - Host Response Time monitor element
- num_transmissions - Number of Transmissions monitor element
- num_transmissions_group - Number of Transmissions Group monitor element
- num_transmissions_group - Number of Transmissions Group monitor element
- con_response_time - Most Recent Response Time for Connect monitor element
- con_elapsed_time - Most Recent Connection Elapsed Time monitor element
- gw_comm_errors - Communication Errors monitor element
- gw_comm_error_time - Communication Error Time monitor element
- blocking_cursor - Blocking Cursor monitor element
- Transaction processor monitoring monitor elements

dcс_db_name - DCS Database Name

Element identifier	dcс_db_name
Element type	information

Table 723. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcс_dbase	Basic
DCS Application	dcс_appl_info	Basic

Description

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

Usage Use this element for problem determination on DCS applications.

Related reference:

- “host_db_name - Host Database Name ” on page 440
- “gw_db_alias - Database Alias at the Gateway ” on page 440

host_db_name - Host Database Name

Element identifier	host_db_name
Element type	information

Table 724. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcс_dbase	Basic
DCS Application	dcс_appl_info	Basic

Description

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.

Usage Use this element for problem determination on DCS applications.

Related reference:

- “dcс_db_name - DCS Database Name ” on page 440
- “gw_db_alias - Database Alias at the Gateway ” on page 440

gw_db_alias - Database Alias at the Gateway

Element identifier	gw_db_alias
Element type	information

Table 725. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcс_appl_info	Basic

Description

The alias used at the DB2 Connect gateway to connect to the host database.

Usage Use this element for problem determination on DCS applications.

Related reference:

- “dcs_db_name - DCS Database Name ” on page 440
- “host_db_name - Host Database Name ” on page 440

gw_con_time - DB2 Connect Gateway First Connect Initiated

Element identifier gw_con_time

Element type timestamp

Table 726. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Timestamp
DCS Application	dcs_appl	Timestamp

Description

The date and time when the first connection to the host database was initiated from the DB2 Connect gateway.

Usage Use this element for problem determination on DCS applications.

gw_connections_top - Maximum Number of Concurrent Connections to Host Database

Element identifier gw_connections_top

Element type water mark

Table 727. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic

Description

The maximum number of concurrent connections to a host database that have been handled by the DB2 Connect gateway since the first database connection.

Usage This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

Related reference:

- “gw_total_cons - Total Number of Attempted Connections for DB2 Connect ” on page 441
- “gw_cur_cons - Current Number of Connections for DB2 Connect ” on page 442

gw_total_cons - Total Number of Attempted Connections for DB2 Connect

Element identifier gw_total_cons

Element type water mark

Table 728. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

DB2 Connect monitor elements

Table 728. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic

For snapshot monitoring, this counter can be reset.

Description

The total number of connections attempted from the DB2 Connect gateway since the last db2start command or the last reset.

Usage This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

Related reference:

- “gw_connections_top - Maximum Number of Concurrent Connections to Host Database ” on page 441
- “gw_cur_cons - Current Number of Connections for DB2 Connect ” on page 442

gw_cur_cons - Current Number of Connections for DB2 Connect

Element identifier gw_cur_cons

Element type gauge

Table 729. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic

Description

The current number of connections to host databases being handled by the DB2 Connect gateway.

Usage This element will help you understand the level of activity at the DB2 Connect gateway and the associated use of system resources.

Related reference:

- “gw_connections_top - Maximum Number of Concurrent Connections to Host Database ” on page 441
- “gw_total_cons - Total Number of Attempted Connections for DB2 Connect ” on page 441

gw_cons_wait_host - Number of Connections Waiting for the Host to Reply

Element identifier gw_cons_wait_host

Element type gauge

Table 730. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic

Description

The current number of connections to host databases being handled by the DB2 Connect gateway that are waiting for a reply from the host.

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.

Related reference:

- “gw_cur_cons - Current Number of Connections for DB2 Connect ” on page 442
- “gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request ” on page 443

gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request

Element identifier gw_cons_wait_client
Element type gauge

Table 731. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic

Description

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.

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.

Related reference:

- “gw_cur_cons - Current Number of Connections for DB2 Connect ” on page 442
- “gw_cons_wait_host - Number of Connections Waiting for the Host to Reply ” on page 442

gw_exec_time - Elapsed Time Spent on DB2 Connect Gateway Processing

Element identifier gw_exec_time
Element type time

Table 732. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Statement, Timestamp
DCS Statement	dcs_stmt	Statement, Timestamp

For snapshot monitoring, this counter can be reset.

Description

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.

DB2 Connect monitor elements

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

Element identifier sql_stmts

Element type counter

Table 733. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic
Data Transmission	stmt_transmissions	Basic

For snapshot monitoring, this counter can be reset.

Description

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.

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.

Notes:

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.

Related reference:

- “num_transmissions_group - Number of Transmissions Group ” on page 469
- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “time_stamp - Snapshot Time ” on page 420

sql_chains - Number of SQL Chains Attempted

Element identifier sql_chains

Element type counter

Table 734. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Basic

For snapshot monitoring, this counter can be reset.

Description

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.

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.

Related reference:

- "num_transmissions_group - Number of Transmissions Group " on page 469
- "sql_stmts - Number of SQL Statements Attempted " on page 444
- "Statement attributes (CLI) list" in *Call Level Interface Guide and Reference, Volume 2*

open_cursors - Number of Open Cursors

Element identifier open_cursors

Element type gauge

Table 735. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Statement

Description

The number of cursors currently open for an application.

Usage Use this element to assess how much memory is being allocated. The amount of memory allocated by the DB2 client, DB2 Connect, or the database agent on the target database is related to the number of cursors that are currently open. Knowing this information can help with capacity planning. For example, each open cursor that is doing blocking has a buffer size of RQRI0BLK. If *deferred_prepare* is enabled, then two buffers will be allocated.

DB2 Connect monitor elements

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

Element identifier dcs_appl_status
Element type information

Table 736. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic

Description

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

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.

Related reference:

- "host_ccsid - Host Coded Character Set ID " on page 447
- "outbound_comm_protocol - Outbound Communication Protocol " on page 447
- "outbound_comm_address - Outbound Communication Address " on page 448
- "inbound_comm_address - Inbound Communication Address " on page 448

agent_status - DCS Application Agents

Element identifier agent_status
Element type information

Table 737. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic

Description

In a connection concentrator environment, this value shows which applications currently have associated agents.

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.

Related reference:

- “dcs_appl_status - DCS Application Status ” on page 446

host_ccsid - Host Coded Character Set ID

Element identifier	host_ccsid
Element type	information

Table 738. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic

Description

This is the coded character set identifier (CCSID) of the host database.

Usage Use this element for problem determination on DCS applications.

Related reference:

- “dcs_appl_status - DCS Application Status ” on page 446
- “outbound_comm_protocol - Outbound Communication Protocol ” on page 447
- “outbound_comm_address - Outbound Communication Address ” on page 448
- “inbound_comm_address - Inbound Communication Address ” on page 448

outbound_comm_protocol - Outbound Communication Protocol

Element identifier	outbound_comm_protocol
Element type	information

Table 739. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic

Description

The communication protocol used between the DB2 Connect gateway and the host.

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

- SQLM_PROT_APPC
- SQLM_PROT_TCPIP

Related reference:

- “dcs_appl_status - DCS Application Status ” on page 446

DB2 Connect monitor elements

- “host_ccsid - Host Coded Character Set ID ” on page 447
- “outbound_comm_address - Outbound Communication Address ” on page 448
- “inbound_comm_address - Inbound Communication Address ” on page 448

outbound_comm_address - Outbound Communication Address

Element identifier	outbound_comm_address
Element type	information

Table 740. Snapshot Monitoring Information

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

Description

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.

Usage Use this element for problem determination on DCS applications.

Related reference:

- “dcs_appl_status - DCS Application Status ” on page 446
- “host_ccsid - Host Coded Character Set ID ” on page 447
- “outbound_comm_protocol - Outbound Communication Protocol ” on page 447
- “inbound_comm_address - Inbound Communication Address ” on page 448

inbound_comm_address - Inbound Communication Address

Element identifier	inbound_comm_address
Element type	information

Table 741. Snapshot Monitoring Information

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

Description

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.

Usage Use this element for problem determination on DCS applications.

Related reference:

- “dcs_appl_status - DCS Application Status ” on page 446
- “host_ccsid - Host Coded Character Set ID ” on page 447
- “outbound_comm_protocol - Outbound Communication Protocol ” on page 447
- “outbound_comm_address - Outbound Communication Address ” on page 448

inbound_bytes_received - Inbound Number of Bytes Received

Element identifier	inbound_bytes_received
Element type	counter

Table 742. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dc_s_appl	Basic
DCS Statement	dc_s_stmt	Statement

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

Description

The number of bytes received by the DB2 Connect gateway from the client, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

Usage Use this element to measure the throughput from the client to the DB2 Connect gateway.

Related reference:

- “outbound_bytes_sent - Outbound Number of Bytes Sent ” on page 449
- “outbound_bytes_received - Outbound Number of Bytes Received ” on page 450
- “inbound_bytes_sent - Inbound Number of Bytes Sent ” on page 450

outbound_bytes_sent - Outbound Number of Bytes Sent

Element identifier	outbound_bytes_sent
Element type	counter

Table 743. Snapshot Monitoring Information

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

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

Description

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.

Usage Use this element to measure the throughput from the DB2 Connect gateway to the host database.

Related reference:

- “inbound_bytes_received - Inbound Number of Bytes Received ” on page 449

DB2 Connect monitor elements

- “outbound_bytes_received - Outbound Number of Bytes Received ” on page 450
- “inbound_bytes_sent - Inbound Number of Bytes Sent ” on page 450

outbound_bytes_received - Outbound Number of Bytes Received

Element identifier outbound_bytes_received
Element type counter

Table 744. 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.

Description

The number of bytes received by the DB2 Connect gateway from the host, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

For the data transmission level: Number of bytes received by the DB2 Connect gateway from the host during the processing of all the statements that used this number of data transmissions.

Usage Use this element to measure the throughput from the host databases to the DB2 Connect gateway.

Related reference:

- “inbound_bytes_received - Inbound Number of Bytes Received ” on page 449
- “outbound_bytes_sent - Outbound Number of Bytes Sent ” on page 449
- “inbound_bytes_sent - Inbound Number of Bytes Sent ” on page 450

inbound_bytes_sent - Inbound Number of Bytes Sent

Element identifier inbound_bytes_sent
Element type counter

Table 745. 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.

Description

The number of bytes sent by the DB2 Connect gateway to the client, excluding communication protocol overhead (for example, TCP/IP or SNA headers).

Usage Use this element to measure the throughput from the DB2 Connect gateway to the client.

Related reference:

- “inbound_bytes_received - Inbound Number of Bytes Received ” on page 449
- “outbound_bytes_sent - Outbound Number of Bytes Sent ” on page 449
- “outbound_bytes_received - Outbound Number of Bytes Received ” on page 450

outbound_bytes_sent_top - Maximum Outbound Number of Bytes Sent

Element identifier outbound_bytes_sent_top
Element type water mark

Table 746. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description

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.

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.

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

outbound_bytes_received_top - Maximum Outbound Number of Bytes Received

Element identifier outbound_bytes_received_top
Element type water mark

Table 747. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description

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.

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.

Related reference:

DB2 Connect monitor elements

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

outbound_bytes_sent_bottom - Minimum Outbound Number of Bytes Sent

Element identifier	outbound_bytes_sent_bottom
Element type	water mark

Table 748. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description

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.

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.

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

outbound_bytes_received_bottom - Minimum Outbound Number of Bytes Received

Element identifier	outbound_bytes_received_bottom
Element type	water mark

Table 749. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description

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.

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.

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_128 - Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes

Element identifier max_data_sent_128

Element type counter

Table 750. 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.

Description

This element represents the number of statements or chains with outbound bytes sent between 1 and 128 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_128 - Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes

Element identifier max_data_received_128

Element type counter

Table 751. 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.

Description

This element represents the number of statements or chains with outbound bytes received between 1 and 128 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_256 - Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes

Element identifier max_data_sent_256

DB2 Connect monitor elements

Element type counter

Table 752. 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.

Description

This element represents the number of statements or chains with outbound bytes sent between 129 and 256 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_256 - Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes

Element identifier max_data_received_256

Element type counter

Table 753. 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.

Description

This element represents the number of statements or chains with outbound bytes received between 129 and 256 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_512 - Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes

Element identifier max_data_sent_512

Element type counter

Table 754. 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.

Description

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

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

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

Element identifier	max_data_received_512
Element type	counter

Table 755. 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.

Description

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

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_1024 - Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes

Element identifier	max_data_sent_1024
Element type	counter

Table 756. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement

DB2 Connect monitor elements

Table 756. 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.

Description

This element represents the number of statements or chains with outbound bytes sent between 513 and 1024 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_1024 - Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes

Element identifier max_data_received_1024

Element type counter

Table 757. 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.

Description

This element represents the number of statements or chains with outbound bytes received between 513 and 1024 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_2048 - Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes

Element identifier max_data_sent_2048

Element type counter

Table 758. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement

Table 758. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Description

This element represents the number of statements or chains with outbound bytes sent between 1025 and 2048 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_2048 - Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes

Element identifier	max_data_received_2048
Element type	counter

Table 759. 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.

Description

This element represents the number of statements or chains with outbound bytes received between 1025 and 2048 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_4096 - Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes

Element identifier	max_data_sent_4096
Element type	counter

Table 760. 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

DB2 Connect monitor elements

For snapshot monitoring, this counter can be reset.

Description

This element represents the number of statements or chains with outbound bytes sent between 2049 and 4096 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_4096 - Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes

Element identifier max_data_received_4096

Element type counter

Table 761. 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.

Description

This element represents the number of statements or chains with outbound bytes received between 2049 and 4096 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_8192 - Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes

Element identifier max_data_sent_8192

Element type counter

Table 762. 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.

Description

This element represents the number of statements or chains with outbound bytes sent between 4097 and 8192 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_8192 - Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes

Element identifier max_data_received_8192

Element type counter

Table 763. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Description

This element represents the number of statements or chains with outbound bytes received between 4097 and 8192 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_16384 - Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes

Element identifier max_data_sent_16384

Element type counter

Table 764. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Description

This element represents the number of statements or chains with outbound bytes sent between 8193 and 16384 inclusive.

DB2 Connect monitor elements

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_16384 - Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes

Element identifier max_data_received_16384

Element type counter

Table 765. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Description

This element represents the number of statements or chains with outbound bytes received between 8193 and 16384 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_31999 - Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes

Element identifier max_data_sent_31999

Element type counter

Table 766. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Description

This element represents the number of statements or chains with outbound bytes sent between 16385 and 31999 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_31999 - Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes

Element identifier max_data_received_31999
 Element type counter

Table 767. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Description

This element represents the number of statements or chains with outbound bytes received between 16385 and 31999 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_sent_64000 - Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes

Element identifier max_data_sent_64000
 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.

Description

This element represents the number of statements or chains with outbound bytes sent between 32000 and 64000 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_64000 - Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes

Element identifier	max_data_received_64000
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.

Description

This element represents the number of statements or chains with outbound bytes received between 32000 and 64000 inclusive.

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

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

Element identifier	max_data_sent_gt64000
Element type	counter

Table 770. 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.

Description

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

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_data_received_gt64000 - Number of Statements with Outbound Bytes Received Greater than 64000 Bytes

Element identifier	max_data_received_gt64000
--------------------	---------------------------

Element type counter

Table 771. 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.

Description

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

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

Related reference:

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444

max_network_time_1_ms - Number of Statements with Network Time of up to 1 ms

Element identifier max_network_time_1_ms

Element type counter

Table 772. 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.

Description

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.)

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

Related reference:

- “host_response_time - Host Response Time ” on page 468
- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

max_network_time_4_ms - Number of Statements with Network Time between 1 and 4 ms

Element identifier max_network_time_4_ms

DB2 Connect monitor elements

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.

Description

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.)

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

Related reference:

- “host_response_time - Host Response Time ” on page 468
- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

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

Element identifier max_network_time_16_ms

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.

Description

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.)

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

Related reference:

- “host_response_time - Host Response Time ” on page 468
- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

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

Element identifier max_network_time_100_ms

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.

Description

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.)

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

Related reference:

- “host_response_time - Host Response Time ” on page 468
- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

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

Element identifier max_network_time_500_ms

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.

Description

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.)

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

Related reference:

- “host_response_time - Host Response Time ” on page 468

DB2 Connect monitor elements

- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

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

Element identifier	max_network_time_gt500_ms
Element type	counter

Table 777. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement
Data Transmission	stmt_transmissions	Statement

For snapshot monitoring, this counter can be reset.

Description

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.)

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

Related reference:

- “host_response_time - Host Response Time ” on page 468
- “sql_chains - Number of SQL Chains Attempted ” on page 444
- “sql_stmts - Number of SQL Statements Attempted ” on page 444
- “total_exec_time - Elapsed Statement Execution Time ” on page 411

network_time_top - Maximum Network Time for Statement

Element identifier	network_time_top
Element type	water mark

Table 778. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement, Timestamp
DCS Application	dcs_appl	Statement, Timestamp
Data Transmission	stmt_transmissions	Statement, Timestamp

For snapshot monitoring, this counter can be reset.

Description

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.)

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.

Related reference:

- “total_exec_time - Elapsed Statement Execution Time ” on page 411
- “host_response_time - Host Response Time ” on page 468

network_time_bottom - Minimum Network Time for Statement

Element identifier network_time_bottom

Element type water mark

Table 779. Snapshot Monitoring Information

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

For snapshot monitoring, this counter can be reset.

Description

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.)

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

Related reference:

- “total_exec_time - Elapsed Statement Execution Time ” on page 411
- “host_response_time - Host Response Time ” on page 468

xid - Transaction ID

Element identifier xid

Element type information

Table 780. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dc_s_appl	Unit of Work

Description

A unique transaction identifier (across all databases) generated by a transaction manager in a two-phase commit transaction.

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

Element identifier elapsed_exec_time

DB2 Connect monitor elements

Element type time

Table 781. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Statement, Timestamp
Application	appl	Statement, Timestamp
DCS Database	dc_s_dbase	Statement, Timestamp
DCS Application	dc_s_appl	Statement, Timestamp
DCS Statement	dc_s_stmt	Statement, Timestamp
Data Transmission	stmt_transmissions	Statement, Timestamp

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

Description

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.

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

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

Note: For the `dc_s_dbase`, `dc_s_appl`, `dc_s_stmt` and `stmt_transmissions` levels, the *elapsed_exec_time element* applies only to z/OS databases. If the DB2 Connect gateway is connecting to a Windows, Linux, AIX, or other UNIX database, the *elapsed_exec_time* is reported as zero.

Related reference:

- "host_response_time - Host Response Time " on page 468

host_response_time - Host Response Time

Element identifier host_response_time

Element type time

Table 782. Snapshot Monitoring Information

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

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

Description

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.

Usage Use this element with Outbound Number of Bytes Sent and Outbound Number of Bytes Received to calculate the outbound response time (transfer rate):

$$(\text{outbound bytes sent} + \text{outbound bytes received}) / \text{host response time}$$
Related reference:

- “outbound_bytes_received - Outbound Number of Bytes Received ” on page 450
- “outbound_bytes_sent - Outbound Number of Bytes Sent ” on page 449
- “elapsed_exec_time - Statement Execution Elapsed Time ” on page 467

num_transmissions - Number of Transmissions

Element identifier num_transmissions

Element type counter

Table 783. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Statement	dcs_stmt	Statement

Description

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.

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.)

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.

Related reference:

- “num_transmissions_group - Number of Transmissions Group ” on page 469

num_transmissions_group - Number of Transmissions Group

Element identifier num_transmissions_group

Element type information

Table 784. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Statement	dcs_stmt	Statement

DB2 Connect monitor elements

Description

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.)

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

Element identifier con_response_time

Element type time

Table 785. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Timestamp

Description

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.

Usage Use this element as an indicator of the time it currently takes applications to connect to a particular host database.

Related reference:

- “pkg_cache_num_overflows - Package Cache Overflows ” on page 279

con_elapsed_time - Most Recent Connection Elapsed Time

Element identifier con_elapsed_time

Element type time

Table 786. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Timestamp

Description

The elapsed time that the DCS application that most recently disconnected from this host database was connected.

Usage Use this element as an indicator of the length of time that applications are maintaining connections to a host database.

Related reference:

- “pkg_cache_num_overflows - Package Cache Overflows ” on page 279

gw_comm_errors - Communication Errors

Element identifier gw_comm_errors

Element type counter

Table 787. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic

For snapshot monitoring, this counter can be reset.

Description

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.

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.

Related reference:

- “gw_comm_error_time - Communication Error Time ” on page 471
- “stmt_elapsed_time - Most Recent Statement Elapsed Time ” on page 392

gw_comm_error_time - Communication Error Time

Element identifier gw_comm_error_time

Element type timestamp

Table 788. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Timestamp

Description

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.

Usage Use this element for problem determination, in conjunction with Communication Error and the communication error logged in administration notification log.

Related reference:

- “gw_comm_errors - Communication Errors ” on page 471

blocking_cursor - Blocking Cursor

Element identifier blocking_cursor

DB2 Connect monitor elements

Element type information

Table 789. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 790. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks with Details	event_detailed_dlconn	-
Statements	event_stmt	-

Description

This element indicates if the statement being executed is using a blocking cursor.

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

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 monitor element
- tpmon_client_wkstn - TP Monitor Client Workstation Name monitor element
- tpmon_client_app - TP Monitor Client Application Name monitor element
- tpmon_acc_str - TP Monitor Client Accounting String monitor element.

tpmon_client_userid - TP Monitor Client User ID

Element identifier tpmon_client_userid

Element type information

Table 791. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
DCS Application	dcs_appl	Basic

Description

The client user ID generated by a transaction manager and provided to the server, if the **sqleseti** API is used.

Usage Use this element in application server or TP monitor environments to identify the end-user for whom the transaction is being executed.

Related reference:

- “tpmon_client_wkstn - TP Monitor Client Workstation Name ” on page 473
- “tpmon_client_app - TP Monitor Client Application Name ” on page 473
- “tpmon_acc_str - TP Monitor Client Accounting String ” on page 474

tpmon_client_wkstn - TP Monitor Client Workstation Name

Element identifier tpmon_client_wkstn

Element type information

Table 792. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
DCS Application	dcs_appl	Basic

Description

Identifies the client’s system or workstation (for example CICS EITERMID), if the **sqleseti** API was issued in this connection.

Usage Use this element to identify the user’s machine by node ID, terminal ID, or similar identifiers.

Related reference:

- “tpmon_client_userid - TP Monitor Client User ID ” on page 472
- “tpmon_client_app - TP Monitor Client Application Name ” on page 473
- “tpmon_acc_str - TP Monitor Client Accounting String ” on page 474

tpmon_client_app - TP Monitor Client Application Name

Element identifier tpmon_client_app

Element type information

Table 793. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
DCS Application	dcs_appl	Basic

Description

Identifies the server transaction program performing the transaction, if the **sqleseti** API was issued in this connection.

Usage Use this element for problem determination and accounting purposes.

Related reference:

- “tpmon_client_userid - TP Monitor Client User ID ” on page 472
- “tpmon_client_wkstn - TP Monitor Client Workstation Name ” on page 473
- “tpmon_acc_str - TP Monitor Client Accounting String ” on page 474

tpmon_acc_str - TP Monitor Client Accounting String

Element identifier tpmon_acc_str

Element type information

Table 794. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic
DCS Application	dcs_appl	Basic

Description

The data passed to the target database for logging and diagnostic purposes, if the `sqleseti` API was issued in this connection.

Usage Use this element for problem determination and accounting purposes.

Related reference:

- “tpmon_client_userid - TP Monitor Client User ID ” on page 472
- “tpmon_client_wkstn - TP Monitor Client Workstation Name ” on page 473
- “tpmon_client_app - TP Monitor Client Application Name ” on page 473

Federated database systems

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. They include:

- `datasource_name` - Data Source Name monitor element
- `disconnects` - Disconnects monitor element
- `insert_sql_stmts` - Inserts monitor element
- `update_sql_stmts` - Updates monitor element
- `delete_sql_stmts` - Deletes monitor element
- `create_nickname` - Create Nicknames monitor element
- `passthru` - Pass-Through monitor element
- `stored_procs` - Stored Procedures monitor element
- `remote_locks` - Remote Locks monitor element
- `sp_rows_selected` - Rows Returned by Stored Procedures monitor element
- `select_time` - Query Response Time monitor element
- `insert_time` - Insert Response Time monitor element
- `update_time` - Update Response Time monitor element
- `delete_time` - Delete Response Time monitor element
- `create_nickname_time` - Create Nickname Response Time monitor element

- passthru_time - Pass-Through Time monitor element
- stored_proc_time - Stored Procedure Time monitor element
- remote_lock_time - Remote Lock Time monitor element

datasource_name - Data Source Name

Element identifier datasource_name
Element type information

Table 795. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase_remote	Basic
Application	appl_remote	Basic

Description

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.

Usage Use this element to identify the data source whose access information has been collected and is being returned.

disconnects - Disconnects

Element identifier disconnects
Element type counter

Table 796. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase_remote	Basic

For snapshot monitoring, this counter can be reset.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier insert_sql_stmts
Element type counter

Table 797. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase_remote	Basic

Federated database systems monitor elements

Table 797. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_remote	Basic

For snapshot monitoring, this counter can be reset.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier	update_sql_stmts
Element type	counter

Table 798. 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.

Description

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 since the later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier	delete_sql_stmts
--------------------	------------------

Element type counter

Table 799. 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.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier create_nickname

Element type counter

Table 800. 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.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier passthru

Federated database systems monitor elements

Element type counter

Table 801. 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.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier stored_procs

Element type counter

Table 802. 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.

Description

This element contains a count of the total number of stored procedures that the federated server has called at this data source on behalf of any application since the later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier remote_locks

Element type counter

Table 803. 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.

Description

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 later of:

- The start of the federated server instance
- The last reset of the database monitor counters.

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

Element identifier sp_rows_selected

Element type counter

Table 804. 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.

Description

This element contains the number of rows sent from the data source to the federated server as a result of stored procedure operations for this application since the later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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:

$$\text{rows per stored procedure} = \frac{\text{rows returned}}{\text{\# of stored procedures invoked}}$$

You can also compute the average time to return a row to the federated server from the data source for this application:

$$\text{average time} = \frac{\text{aggregate stored proc. response time}}{\text{rows returned}}$$

select_time - Query Response Time

Element identifier select_time

Element type counter

Table 805. 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.

Description

This element contains the aggregate amount of time, in milliseconds, that it

Federated database systems monitor elements

has taken this data source to respond to queries from all applications or a single application running on this federated server instance since the later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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.

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.

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.

insert_time - Insert Response Time

Element identifier insert_time
Element type counter

Table 806. 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.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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.

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

Element identifier update_time
Element type counter

Table 807. 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.

Description

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 since the later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Element identifier	delete_time
Element type	counter

Table 808. 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.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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.

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

Element identifier	create_nickname_time
--------------------	----------------------

Federated database systems monitor elements

Element type counter

Table 809. 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.

Description

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 since the later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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.

Usage Use this element to determine how much actual time was used to create nicknames for this data source.

passthru_time - Pass-Through Time

Element identifier passthru_time

Element type counter

Table 810. 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.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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.

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

Element identifier stored_proc_time

Element type counter

Table 811. 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.

Description

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 since the later of:

- The start of the federated server instance
- The last reset of the database monitor counters.

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

Element identifier	remote_lock_time
Element type	counter

Table 812. 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.

Description

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 later of:

- The start of the federated server instance, or
- The last reset of the database monitor counters.

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

Usage Use this element to determine how much actual time is spent at this data source in a remote lock.

Federated database systems monitor elements

Chapter 7. Monitor Interfaces

Database system monitor interfaces

Monitoring task	API
Capturing a snapshot	db2GetSnapshot API - Get a snapshot of the database manager operational status db2GetSnapshot API - Get a snapshot of the database manager operational status
Converting the self-describing data stream	db2ConvMonStream API - Convert the monitor stream to the pre-version 6 format db2ConvMonStream API - Convert the monitor stream to the pre-version 6 format
Displaying the database system monitor switches	db2MonitorSwitches API - Get or update the monitor switch settings
Estimating the size of a snapshot	db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API
Get/update monitor switches	db2MonitorSwitches API - Get or update the monitor switch settings db2MonitorSwitches API - Get or update the monitor switch settings
Resetting monitor counters	db2ResetMonitor API - Reset the database system monitor data db2ResetMonitor API - Reset the database system monitor data
Updating the database system monitor switches	db2MonitorSwitches API - Get or update the monitor switch settings

Monitoring task	CLP Command
Analyzing event monitor output with a GUI tool	db2eva - Event analyzer command db2eva - Event analyzer command
Capturing a snapshot	GET SNAPSHOT command GET SNAPSHOT command
Displaying the database manager monitor switches	GET DATABASE MANAGER MONITOR SWITCHES command GET DATABASE MANAGER MONITOR SWITCHES command
Displaying the monitoring application's monitor switches	GET MONITOR SWITCHES command GET MONITOR SWITCHES command
Formatting the event monitor trace	db2evmon - Event monitor productivity tool command db2evmon - Event monitor productivity tool command
Generating sample SQL for write-to-table CREATE EVENT MONITOR statements	db2evtbl
Listing the active databases	LIST ACTIVE DATABASES command LIST ACTIVE DATABASES command
Listing the applications connected to a database	LIST APPLICATIONS command LIST APPLICATIONS command
Listing the DCS applications	LIST DCS APPLICATIONS command LIST DCS APPLICATIONS command
Resetting monitor counters	RESET MONITOR command RESET MONITOR command
Updating the database system monitor switches	UPDATE MONITOR SWITCHES command UPDATE MONITOR SWITCHES command

Database system monitor interfaces

Monitoring task	SQL Statement
Activating an event monitor	SET EVENT MONITOR STATE statement SET EVENT MONITOR STATE statement
Creating an event monitor	CREATE EVENT MONITOR statement CREATE EVENT MONITOR statement
Deactivating an event monitor	SET EVENT MONITOR STATE statement SET EVENT MONITOR STATE statement
Removing an event monitor	DROP statement DROP statement
Writing event monitor values	FLUSH EVENT MONITOR statement FLUSH EVENT MONITOR statement

Monitoring task	SQL Function
Determining the state of an event monitor	EVENT_MON_STATE scalar function EVENT_MON_STATE scalar function
Getting a database manager level snapshot	SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information
Getting the current monitor switch settings at the database manager level	SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information
Getting a fast communication manager snapshot	SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information
Getting a fast communication manager snapshot for a given partition	SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information
Getting a database level snapshot	SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group
Getting an application level snapshot	SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information
Getting an application level snapshot	SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information
Getting an application level snapshot for lock wait information	SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information

Monitoring task	SQL Function
Getting an application level snapshot for statement information	SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information
Getting an application level snapshot for agent information	SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information
Getting an application level snapshot for subsection information	SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information
Getting a buffer pool level snapshot	SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information
Getting a table space level snapshot	SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information
Getting a table space level snapshot for configuration information	SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information
Getting a table space level snapshot for container information	SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information
Getting a table space level snapshot for quiescer information	SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function – Retrieve quiescer table space snapshot information SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function – Retrieve quiescer table space snapshot information
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 – Retrieve range snapshot information SNAPTbsp_RANGE administrative view and SNAP_GET_TBSP_RANGE table function – Retrieve range snapshot information
Getting a table level snapshot	SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information

Database system monitor interfaces

Monitoring task	SQL Function
Getting a lock level snapshot	SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information
Getting a snapshot of SQL statement cache information	SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information

Related reference:

- “CREATE EVENT MONITOR statement” in *SQL Reference, Volume 2*
- “db2ConvMonStream API - Convert the monitor stream to the pre-version 6 format” in *Administrative API Reference*
- “db2eva - Event analyzer command” in *Command Reference*
- “db2evmon - Event monitor productivity tool command” in *Command Reference*
- “db2GetSnapshot API - Get a snapshot of the database manager operational status” in *Administrative API Reference*
- “db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API” in *Administrative API Reference*
- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*
- “db2ResetMonitor API - Reset the database system monitor data” in *Administrative API Reference*
- “EVENT_MON_STATE scalar function” in *SQL Reference, Volume 1*
- “FLUSH EVENT MONITOR statement” in *SQL Reference, Volume 2*
- “GET DATABASE MANAGER MONITOR SWITCHES command” in *Command Reference*
- “GET MONITOR SWITCHES command” in *Command Reference*
- “GET SNAPSHOT command” in *Command Reference*
- “LIST ACTIVE DATABASES command” in *Command Reference*
- “LIST APPLICATIONS command” in *Command Reference*
- “LIST DCS APPLICATIONS command” in *Command Reference*
- “RESET MONITOR command” in *Command Reference*
- “SET EVENT MONITOR STATE statement” in *SQL Reference, Volume 2*
- “SYSCAT.EVENTMONITORS catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.EVENTS catalog view” in *SQL Reference, Volume 1*

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 5 describes the process for using the Activity monitor to solve a problem.

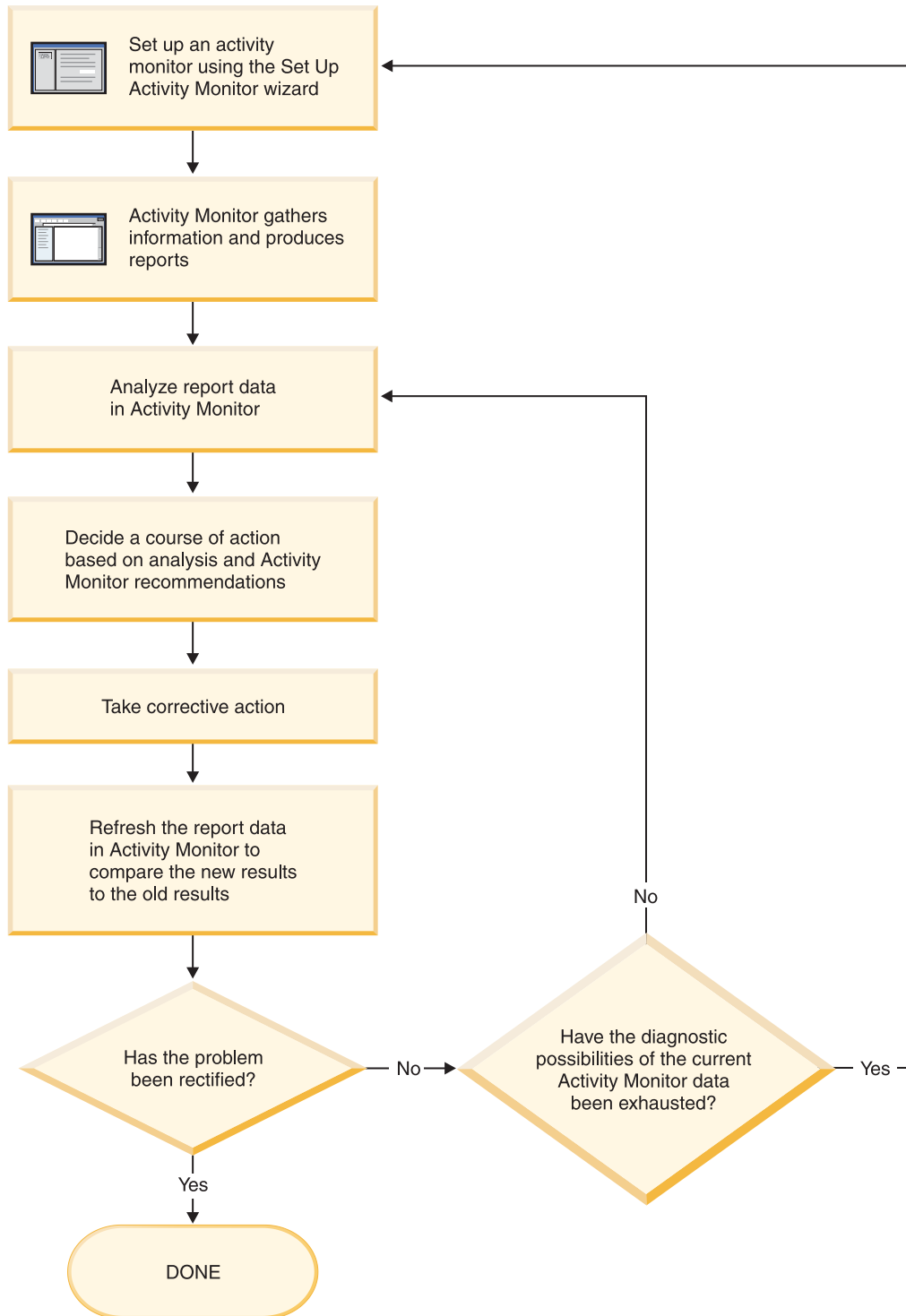


Figure 5. Activity Monitor overview

Database system monitor interfaces

Prerequisites:

When you want to monitor database activity or solve a particular problem using the Activity Monitor, your first step will generally be to invoke the Set Up Activity Monitor wizard. For more information, see [Setting up an activity monitor](#).

You can have more than one Activity Monitor window open at one time.

Tasks from the activity monitor:

Table 813. 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.

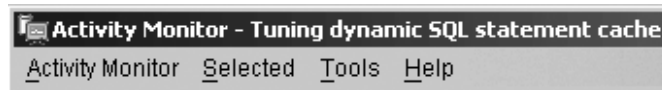
Table 813. Tasks that you can perform from the Activity Monitor (continued)

Tasks from the Activity Monitor	Aspects of tasks	Invocation
	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:

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

Report data					
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	acmeffin.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.

Database system monitor interfaces

Related tasks:

- “Setting up an activity monitor” on page 492

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.

Prerequisites:

To use the Activity Monitor:

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

Procedure:

You can set up an activity monitor through the Set Up Activity Monitor wizard in the Control Center. To open the the Set Up Activity Monitor wizard from the command line, type the following command: **db2am**. The Set Up Activity Monitor wizard opens.

To open the Set Up Activity Monitor wizard from the Control Center:

1. Expand the object tree until you find the instance or the database for which you want to set up an activity monitor.
2. Right-click on the object and select Set Up Activity Monitor from the pop-up menu. The Set Up Activity Monitor wizard opens.

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

Related concepts:

- “Activity Monitor overview” on page 488

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.

To open the Memory Visualizer from the Windows **Start** menu: Click **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**.

To open the Memory Visualizer from the Control Center: Expand the object tree until you find the instance that you want to view. Right-click the instance and select **View memory usage** from the pop-up menu.

Tasks from the Memory Visualizer:

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.

For details about performing the above tasks, see *Working with the Memory Visualizer*.

The Memory Visualizer interface:

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%.

Database system monitor interfaces

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 *Working with the Memory Visualizer*.

Accessing custom controls with the keyboard:

You can use the keyboard to access controls found on the user interface.

For more information, see *Keyboard shortcuts and accelerators (all centers)*.

Related tasks:

- “Working with the Memory Visualizer” on page 495

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.

You can also 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.

Database system monitor interfaces

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

To view memory performance using the Memory Visualizer::

1. 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.
2. To display a plotted graph of a memory component, use one of the following methods:
 - a. Select a component in the hierarchical tree and click the **Show Plot** check box in the Memory Visualizer window.
 - b. Right-click the selected memory component to display a pop-up menu and select **Show Plot**.
 - c. 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.
 - d. 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.

3. 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.
4. 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.
5. 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.

6. 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.

Related concepts:

- “Memory Visualizer overview” on page 492

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.

Database system monitor interfaces

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

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

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

```
db2indbt
```

Tasks from the Indoubt Transaction Manager:

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 Indoubt Transaction manager interface:

The columns in the Indoubt Transaction Manager window provide named views that you can use to organize and display indoubt transactions in different ways.

Columns in the Indoubt Transaction Manager window:

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.

Accessing custom controls with the keyboard:

You can use the keyboard to access controls found on the user interface.

For more information, see Keyboard shortcuts and accelerators (all centers).

Database system monitor interfaces

Related reference:

- “LIST DRDA INDOUBT TRANSACTIONS command” in *Command Reference*
- “LIST INDOUBT TRANSACTIONS command” in *Command Reference*

Part 3. Health Monitor Guide

Chapter 8. Introducing the health monitor

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.

Related concepts:

- “Health indicator process cycle” on page 505
- “Health monitor” on page 511

Related tasks:

- “Enabling health alert notification” on page 507

Related reference:

- “Health indicators” on page 503

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 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

Introducing the health monitor

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 Web 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 and alarm thresholds of 80% and 90% respectively.

Table 814. 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

Table 814. Health indicator conditions at different refresh intervals (continued)

Refresh interval	Value of ts.ts_util (Table space utilization) health indicator (%)	State of ts.ts_util health indicator	Health monitor response
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

Related concepts:

- “Health monitor” on page 511

Related reference:

- “Health indicator format” on page 533
- “Health indicators summary” on page 533

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.

Introducing the health monitor

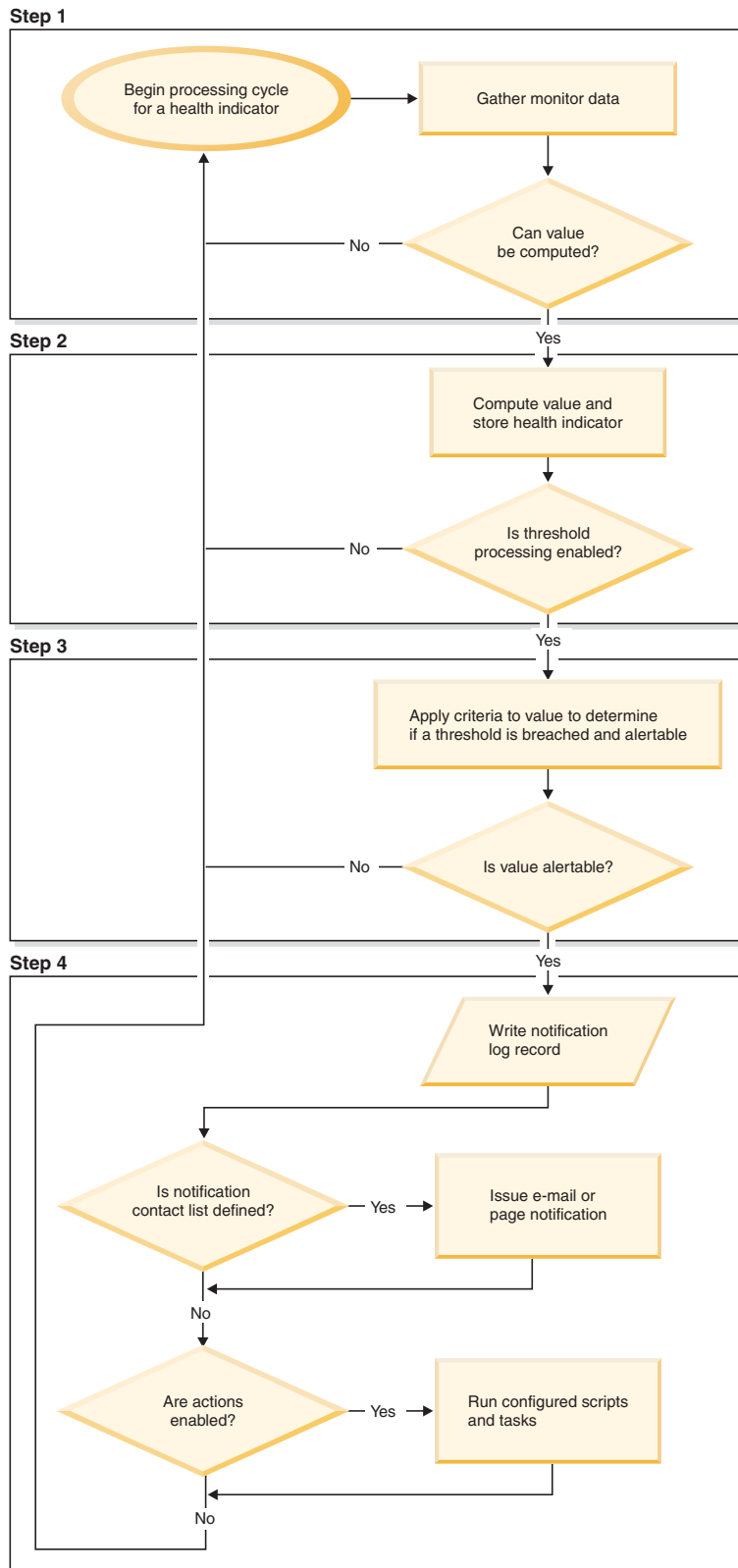


Figure 6. Health indicator process cycle

Notes:

1. The NOTIFYLEVEL database manager configuration parameter controls whether alert notifications are sent to the DB2 administration notification log

and to any defined contacts. A minimum severity level of 2 is required for alarm notifications. A minimum severity level of 3 is required for warnings and attention alerts to be sent.

2. When migrating a Version 7 installation of the DB2 database on Windows, the value of the NOTIFYLEVEL database manager configuration parameter is not updated.

Related reference:

- “notifylevel - Notify level configuration parameter” in *Performance Guide*
- “UPDATE DATABASE MANAGER CONFIGURATION command” in *Command Reference*

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.

Prerequisite:

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.

Procedure:

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:

Introducing the health monitor

```
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.

Related reference:

- “ADD CONTACT command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*
- “UPDATE HEALTH NOTIFICATION CONTACT LIST command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*
- “ADD CONTACT command” in *Command Reference*
- “UPDATE ADMIN CONFIGURATION command” in *Command Reference*
- “UPDATE HEALTH NOTIFICATION CONTACT LIST command” in *Command Reference*

Introducing the health monitor

Chapter 9. Using the health monitor

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.

On Windows, the health monitor process is called DB2FMP.

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.

Using the health monitor

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.

Related concepts:

- “Introduction to the health monitor” on page 503

Related tasks:

- “Capturing a database health snapshot from a client application” on page 516
- “Capturing a database health snapshot using SQL table functions” on page 513
- “Capturing a database health snapshot using the CLP” on page 515

Related reference:

- “Global health snapshots” on page 522
- “Health monitor sample output” on page 520
- “Health monitor SQL table functions” on page 514

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.

Related reference:

- “Database Highest Severity Alert State ” on page 546
- “Instance Highest Severity Alert State ” on page 545

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.

Procedure:

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.

Exception:

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
```

Related concepts:

- “Health monitor” on page 511

Related reference:

- “Health monitor interfaces” on page 559
- “Health monitor SQL table functions” on page 514

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 815. 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

Related concepts:

- “Health monitor” on page 511

Related tasks:

- “Capturing a database health snapshot using the CLP” on page 515

Related reference:

- “Supported administrative SQL routines and views” in *Administrative SQL Routines and Views*
- “Health monitor interfaces” on page 559

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.

Prerequisite:

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.

Procedure:

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:

```
db2 get health snapshot for db on sample with full collection
```

Related concepts:

- “Health monitor” on page 511

Related reference:

- “GET HEALTH SNAPSHOT command” in *Command Reference*
- “Global health snapshots” on page 522
- “Health monitor CLP commands” on page 516
- “Health monitor sample output” on page 520

Health monitor CLP commands

The following table lists all the supported snapshot request types.

Table 816. Snapshot monitor CLP commands

Monitor level	CLP command	Information returned
Database manager	get health snapshot for dbm	Database manager level information.
Database	get health snapshot for all databases	Database level information. Information is returned only if there is at least one application connected to the database.
Database	get health snapshot for database on <i>database-alias</i>	Database level information. Information is returned only if there is at least one application connected to the database.
Database	get health snapshot for all on <i>database-alias</i>	Database, table space, and table space container information. Information is returned only if there is at least one application connected to the database.
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.

Related tasks:

- “Capturing a database health snapshot using the CLP” on page 515

Related reference:

- “GET HEALTH SNAPSHOT command” in *Command Reference*
- “Health monitor interfaces” on page 559

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.

Prerequisites:

You must be attached to an instance to capture a health snapshot. If there is not an attachment to an instance, then a default instance attachment is created. To obtain a snapshot of a remote instance, you must first attach to that instance.

Procedure:

1. Include the sqlmon.h and db2ApiDf.h DB2 libraries in your code. These libraries are found in the sqllib\include directory.

```
#include <db2ApiDf.h>
#include <sqlmon.h>
```
2. Set the snapshot buffer unit size to 50 KB.

```
#define SNAPSHOT_BUFFER_UNIT_SZ 51200
```
3. Declare the sqlma, sqlca, sqlm_collected, and db2GetSnapshotData structures.

```
struct sqlma *pRequestedDataGroups;
struct sqlca sqlca;
memset (&sqlca, '\0', sizeof(struct sqlca));
```

```

struct sqlm_collected collectedData;
memset (&sqlm_collected, '\0', sizeof(struct sqlm_collected));
db2GetSnapshotData getSnapshotParam;
memset(&db2GetSnapshotData, '\0', sizeof(db2GetSnapshotData));

```

4. Initialize a pointer to contain the snapshot buffer, and to establish the buffer's size.

```

static sqluint32 snapshotBufferSize = SNAPSHOT_BUFFER_UNIT_SZ;
sqluint32 outputFormat;
char *snapshotBuffer;

```

5. Initialize the sqlma structure and specify that the snapshot you are capturing is of database manager level information.

```

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

```

6. Initialize the buffer, which will hold the snapshot output.

```

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

```

7. Populate the db2GetSnapshotData structure with the snapshot request type (from the sqlma structure), buffer information, and other information required to capture a snapshot.

```

getSnapshotParam.piSqlmaData = pRequestedDataGroups;
getSnapshotParam.poCollectedData = &collectedData;
getSnapshotParam.poBuffer = snapshotBuffer;
getSnapshotParam.iVersion = SQLM_DBMON_VERSION8;
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.

Using the health monitor

```
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:

Legend:

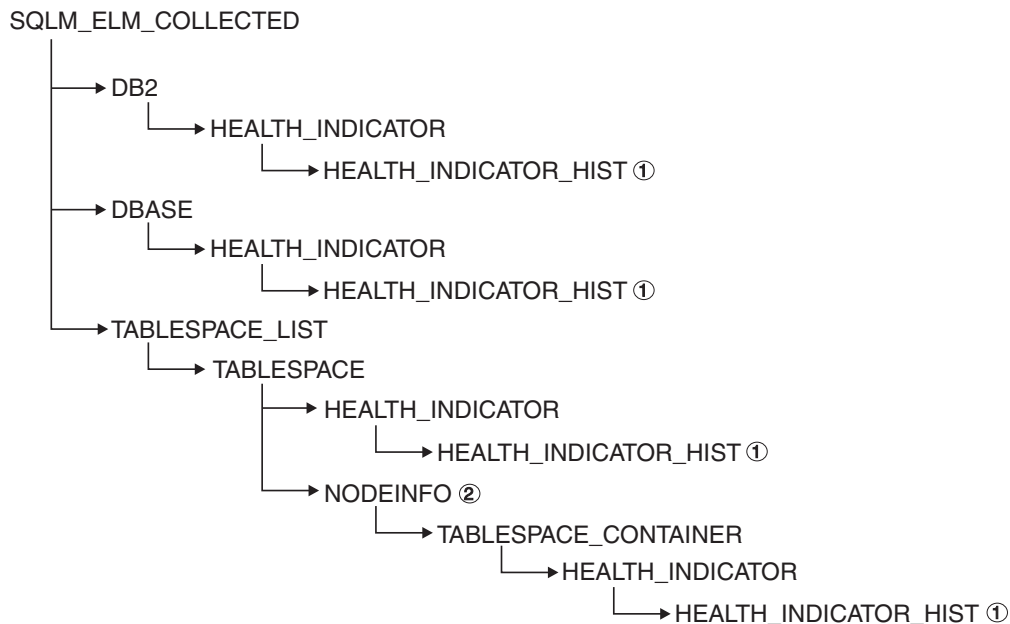


Figure 7. Health snapshot self-describing data stream

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
```

Related concepts:

- “Health monitor” on page 511
- “Snapshot monitor self-describing data stream” on page 54
- “System monitor output: the self-describing data stream” on page 8

Related reference:

- “db2GetSnapshot API - Get a snapshot of the database manager operational status” in *Administrative API Reference*
- “db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API” in *Administrative API Reference*
- “Health monitor API request types” on page 519

Health monitor API request types

The following table lists all the supported snapshot request types.

Table 817. 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 there is at least one application connected to the database.

Using the health monitor

Table 817. Snapshot Monitor API Request Types (continued)

Monitor level	API request type	Information returned
Database	SQLMA_DBASE	Database level information. Information is returned only if there is at least one application connected to the database.
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.

Related concepts:

- “Snapshot monitor self-describing data stream” on page 54

Related tasks:

- “Capturing a database health snapshot from a client application” on page 516

Related reference:

- “db2GetSnapshot API - Get a snapshot of the database manager operational status” in *Administrative API Reference*
- “db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API” in *Administrative API Reference*

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.

Example 1 Procedure:

1. Take the database manager snapshot, using the GET HEALTH SNAPSHOT command:

```
db2 get health snapshot for dbm
```

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

```
Node name =  
Node type = Database Server with local  
and remote clients  
Instance name = DB2  
Snapshot timestamp = 11-07-2002 12:43:23.613425  
  
Number of database partitions in DB2 instance = 1  
Start Database Manager timestamp = 11-07-2002 12:43:18.000108  
Instance highest severity alert state = Not yet evaluated
```

Health Indicators:

```
Not yet evaluated
```

2. Analyze the output.

From this health snapshot, you can see that the instance highest severity alert state is “Not yet evaluated”. The instance is in this state because the health monitor has just started and has not yet evaluated any health indicators.

Should the instance highest severity alert state not change:

- Check the value of the HEALTH_MON database manager configuration parameter to determine if the health monitor is on.
- If HEALTH_MON=OFF, then the health monitor is not started. To start the health monitor, issue the UPDATE DBM CFG USING HEALTH_MON ON command.
- If HEALTH_MON=ON, attach to the instance to activate the health monitor. If an instance attachment exists, it is possible that the health monitor could not be loaded into memory.

Another example of taking a database health snapshot using the CLP is outlined below.

Example 2 Prerequisite:

- A database connection must exist.
- The database must be quiesced.

Example 2 Procedure:

1. 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

```

2. 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.

Related concepts:

- “Health monitor” on page 511

Related reference:

- “Health monitor CLP commands” on page 516

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 818. 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>
<ul style="list-style-type: none"> • db.tb_reorg_req • db.tb_runstats_req • db.fed_nicknames_op_status • db.fed_servers_op_status 	<p>This health indicator is evaluated only on one partition, so no aggregation is required. The data is returned from the partition which is evaluating the health indicator.</p>

Note: When taking a global snapshot on a single partition object, the output includes all the attributes because there are no partitions to aggregate.

Related concepts:

- “Health monitor” on page 511

Related tasks:

- “Capturing a database health snapshot using the CLP” on page 515
- “Capturing a database health snapshot using SQL table functions” on page 513
- “Capturing a database health snapshot from a client application” on page 516

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.

Using the health monitor

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.

Web Health Center

The Web Health Center is accessible through a Web browser or a Web-enabled personal digital assistant (PDA). The interface that is presented to the user differs depending on which medium is being used, but the content is the same. The Web Health Center is intended for users who would normally use the full Health Center, but are currently away from their usual point of access.

The Web Health Center provides functionality to view the health information for a particular instance and all of its children objects. You cannot choose a database-specific context through the Web Health Center.

The first step in using the Web Health Center is to log on to the application at the DB2 Web Tools site that was set up during installation. From the main page, click the Health Center tab (or link on the PDA). First, you will be prompted to select a system and to provide your userid and password for authentication. Next, you must select the instance whose health status you want to view.

The DB2 Web tools can automatically catalog systems, instances, and databases on your network using DB2 Discovery. If you choose to disable these automatic cataloging options by turning the flags to false in web.xml, you must manually catalog the systems, instances, and databases on the applications server.

When you select an instance, the Web Health Center opens the Current Alerts page. This page lists all the alerts that currently exist on the selected instance, any of its catalogued databases, and any table space and table space containers in those databases. Full details of the alert as described for the Health Center are available by clicking on an alert.

The Web Health Center in a PDA displays the details of the health indicator on the Description view, which is the first screen accessible from the Current Alerts view. The health indicator recommendations and history are accessible through links at the top of the Description view.

Health Center overview

Use the Health Center to analyse 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 DB2.

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
```

Tasks from the Health Center:

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

- Enabling health alert notification
 - Specifying contact settings and notification configuration parameters
 - Troubleshooting health alert notification

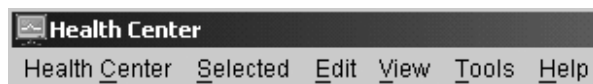
Using the health monitor

- Configuring health indicators using the Health Center
 - Enabling and disabling health indicator evaluation
 - Changing alert threshold and sensitivity settings
 - Running tasks and scripts when an alert occurs
- Resolving alerts using the Health Center
 - 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 menubar 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

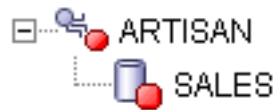


Shows objects in any alert state: alarm, warning, attention, normal, and not monitored.



Shows all objects

Navigation view

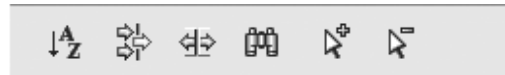


Use the navigation view to display and work with instance and database objects. When you select an object in the Navigation view, the current alerts for that object and all its children are displayed in the Alerts view. To change the level of alert that object must have before the navigation view displays 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.

Accessing custom controls with the keyboard:

You can use the keyboard to access controls found on the user interface.

For more information, see Keyboard shortcuts and accelerators (all centers).

Part 4. Health Monitor Reference

Chapter 10. Health Monitor Logical Data Groups

Health monitor interface mappings to logical data groups

The following table lists all the supported health snapshot request types.

Table 819. 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.

Health Monitor Logical Data Groups

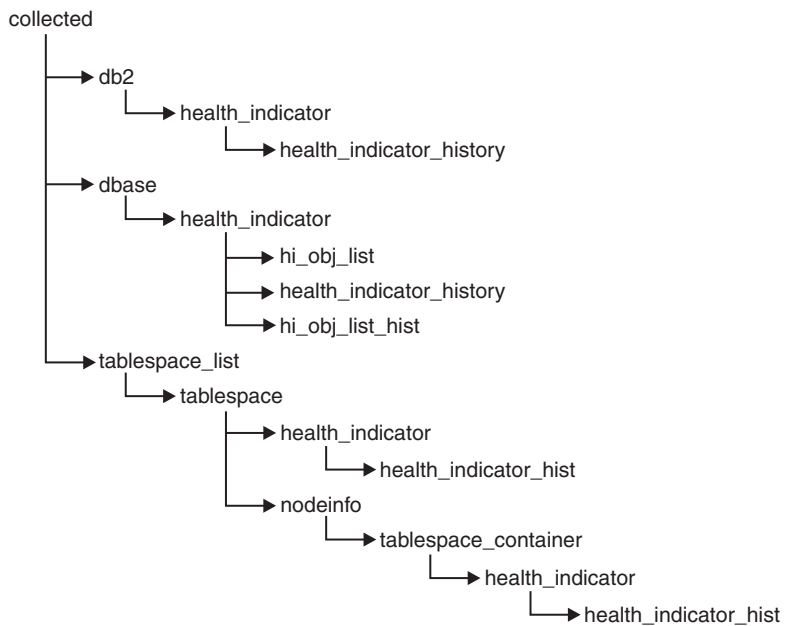


Figure 8. Health snapshot logical data groupings

Related reference:

- “GET HEALTH SNAPSHOT command” in *Command Reference*
- “Health monitor interfaces” on page 559

Chapter 11. Health Indicators

Health indicator format

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: <ul style="list-style-type: none">• 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.
Description	A description of the data collected by the health indicator.

Related concepts:

- “Database system monitor” on page 3
- “Database system monitor data organization” on page 3

Related reference:

- “Health indicators” on page 503

Health indicators summary

The following tables list all health indicators, grouped by category.

Table 820. 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 536

Health indicators

Table 821. 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 537
Automatic Resize Status Table Space Utilization	ts.ts_util_auto_resize	"ts.ts_util_auto_resize – Automatic resize table space utilization health indicator" on page 538
Table Space Utilization	ts.ts_util	"ts.ts_util - Table Space Utilization " on page 538
Table Space Container Utilization	tsc.tscont_util	"tsc.tscont_util - Table Space Container Utilization " on page 539
Table Space Operational State	ts.ts_op_status	"ts.ts_op_status - Table Space Operational State " on page 540
Table Space Container Operational State	tsc.tscont_op_status	"tsc.tscont_op_status - Table Space Container Operational State " on page 541
Table Space Automatic Resize Status	ts.ts_auto_resize_status	"ts.ts_auto_resize_status – Table space automatic resize status health indicator" on page 537

Table 822. 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 541
Shared Sort Memory Utilization	db.sort_shrmem_util	"db.sort_shrmem_util - Shared Sort Memory Utilization " on page 542
Percentage of Sorts That Overflowed	db.spilled_sorts	"db.spilled_sorts - Percentage of Sorts That Overflowed " on page 543
Long Term Shared Sort Memory Utilization	db.max_sort_shrmem_util	"db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization " on page 544

Table 823. Database manager health indicators

Name	Identifier	Additional Information
Instance Operational State	db2.db2_op_status	"db2.db2_op_status - Instance Operational State " on page 544
Instance Highest Severity Alert State	–	"Instance Highest Severity Alert State " on page 545

Table 824. Database health indicators

Name	Identifier	Additional Information
Database Operational State	db.db_op_status	"db.db_op_status - Database Operational State " on page 546
Database Highest Severity Alert State	–	"Database Highest Severity Alert State " on page 546

Table 825. Maintenance health indicators

Name	Identifier	Additional Information
Reorganization Required	db.tb_reorg_req	"db.tb_reorg_req - Reorganization Required " on page 546
Statistics Collection Required health indicator	db.tb_runstats_req	"db.tb_runstats_req - Statistics Collection Required " on page 547
Database Backup Required	db.db_backup_req	"db.db_backup_req - Database Backup Required " on page 548

Table 826. 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 548
HADR Log Delay health indicator	db.hadr_delay	"db.hadr_delay - HADR Log Delay " on page 549

Table 827. Logging health indicators

Name	Identifier	Additional Information
Log Utilization	db.log_util	"db.log_util - Log Utilization" on page 549
Log Filesystem Utilization	db.log_fs_util	"db.log_fs_util - Log Filesystem Utilization" on page 550

Table 828. Application concurrency health indicators

Name	Identifier	Additional Information
Deadlock Rate	db.deadlock_rate	"db.deadlock_rate - Deadlock Rate" on page 550
Lock List Utilization	db.locklist_util	"db.locklist_util - Lock List Utilization " on page 551
Lock Escalation Rate	db.lock_escal_rate	"db.lock_escal_rate - Lock Escalation Rate" on page 552
Percentage of Applications Waiting on Locks	db.apps_waiting_locks	"db.apps_waiting_locks - Percentage of Applications Waiting on Locks" on page 553

Table 829. Package and catalog caches, and workspaces health indicators

Name	Identifier	Additional Information
Catalog Cache Hit Ratio	db.catcache_hitratio	"db.catcache_hitratio - Catalog Cache Hit Ratio" on page 553
Package Cache Hit Ratio	db.pkgcache_hitratio	"db.pkgcache_hitratio - Package Cache Hit Ratio " on page 554
Shared Workspace Hit Ratio	db.shrworkspace_hitratio	"db.shrworkspace_hitratio - Shared Workspace Hit Ratio " on page 554

Table 830. Memory health indicators

Name	Identifier	Additional Information
Monitor Heap Utilization	db2.mon_heap_util	"db2.mon_heap_util - Monitor Heap Utilization " on page 555
Database Heap Utilization	db.db_heap_util	"db.db_heap_util - Database Heap Utilization " on page 555

Table 831. Federated health indicators

Name	Identifier	Additional Information
Nickname Status	db.fed_nicknames_op_status	"db.fed_nicknames_op_status - Nickname Status " on page 556
Data Source Server Status	db.fed_servers_op_status	"db.fed_servers_op_status - Data Source Server Status " on page 556

Related reference:

- "Health indicators" on page 503

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:

Health indicators

Table 832. 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 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	Not a valid configuration. Maximum table space size is only valid for table spaces that have automatic resize enabled.	<p>ts.ts_util_auto_resize - Not applicable. Table space will not attempt to resize.</p> <p>ts.ts_util - Tracks usage of currently allocated table space storage. An alert indicates a table space full condition and requires immediate intervention by you. The table space will not attempt to resize itself.</p> <p>ts.ts_auto_resize_status - Not applicable. Table space will not attempt to resize.</p>

Database storage health indicators

db.auto_storage_util – Database automatic storage utilization health indicator

Identifier	db.auto_storage_util
Health monitor level	Database
Category	Database
Type	Upper-bounded threshold-based
Unit	Percentage

Description

This health indicator tracks the consumption of storage for the defined database storage paths. 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* and *db.auto_storage_total* are the sum of used and total space respectively 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.

Related concepts:

- “Automatic storage databases” in *Administration Guide: Implementation*

Related reference:

- “Database Highest Severity Alert State ” on page 546

Table space storage health indicators

ts.ts_auto_resize_status – Table space automatic resize status health indicator

Identifier	ts.ts_auto_resize_status
Health monitor level	Table Space
Category	Table Space Storage
Type	State-based
Unit	Not applicable

Description

This health indicator identifies whether or not 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.

Related reference:

- “db.auto_storage_util – Database automatic storage utilization health indicator” on page 536

Health indicators

- “Health indicators for DMS table spaces” on page 535
- “tablespace_last_resize_failed - Last resize attempt failed ” on page 337
- “ts.ts_util - Table Space Utilization ” on page 538
- “tsc.tscont_op_status - Table Space Container Operational State ” on page 541
- “tsc.tscont_util - Table Space Container Utilization ” on page 539

ts.ts_util_auto_resize – Automatic resize table space utilization health indicator

Identifier	ts.ts_util_auto_resize
Health monitor level	Table Space
Category	Table Space Storage
Type	Upper-bounded threshold-based
Unit	Percentage

Description

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.

The indicator is calculated using the formula:

$$((ts.used * ts.page_size) / ts.max_size) * 100$$

where *ts.used*, *ts.page_size*, and *ts.max_size* are the system monitor data elements Used Pages in Table Space, Table Space Page Size, and Maximum Table Space Size, respectively.

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.

Related reference:

- “Health indicators for DMS table spaces” on page 535
- “tablespace_current_size - Current table space size ” on page 335
- “tablespace_max_size - Maximum table space size ” on page 336
- “ts.ts_util - Table Space Utilization ” on page 538
- “tsc.tscont_op_status - Table Space Container Operational State ” on page 541
- “tsc.tscont_util - Table Space Container Utilization ” on page 539
- “ts.ts_auto_resize_status – Table space automatic resize status health indicator” on page 537

ts.ts_util - Table Space Utilization

Identifier	ts.ts_util
-------------------	------------

Health monitor level	Table Space
Category	Table Space Storage
Type	Upper-bounded threshold-based
Unit	Percentage

Description

This health indicator tracks the consumption of storage for each DMS table space.

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 tablespace automatic resize status (*ts.ts_auto_resize_status*) health indicators are relevant for table space storage monitoring. The automatic resize tablespace utilization (*ts.ts_util_auto_resize*) health indicator will also be available if a maximum size was defined on this table space. The tablespace 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:

$$(ts.used / ts.useable) * 100$$

where *ts.used* and *ts.useable* are the system monitor elements *Used Pages in Table Space* and *Useable Pages in Table Space*, respectively.

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.

Related reference:

- “ts.ts_op_status - Table Space Operational State ” on page 540
- “ts.ts_util_auto_resize – Automatic resize table space utilization health indicator” on page 538
- “Health indicators” on page 503
- “Health indicators for DMS table spaces” on page 535
- “tsc.tscont_op_status - Table Space Container Operational State ” on page 541
- “tsc.tscont_util - Table Space Container Utilization ” on page 539
- “ts.ts_auto_resize_status – Table space automatic resize status health indicator” on page 537

tsc.tscont_util - Table Space Container Utilization

Identifier	tsc.tscont_util
Health monitor level	Table Space Container

Health indicators

Category	Table Space Storage
Type	Upper-bounded threshold-based
Unit	Percentage

Description

This health indicator tracks the consumption of storage for each SMS table space that is not using automatic storage. 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.

Related reference:

- “ts.ts_util_auto_resize – Automatic resize table space utilization health indicator” on page 538
- “Health indicators” on page 503
- “ts.ts_op_status - Table Space Operational State ” on page 540
- “ts.ts_util - Table Space Utilization ” on page 538
- “tsc.tscont_op_status - Table Space Container Operational State ” on page 541
- “ts.ts_auto_resize_status – Table space automatic resize status health indicator” on page 537

ts.ts_op_status - Table Space Operational State

Identifier	ts.ts_op_status
Health monitor level	Table Space
Category	Table Space Storage
Type	State-based
Unit	Not applicable

Description

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.

Related reference:

- “Health indicators” on page 503
- “LIST TABLESPACES command” in *Command Reference*
- “tablespace_state - Table Space State ” on page 330
- “tsc.tscont_op_status - Table Space Container Operational State ” on page 541
- “tsc.tscont_util - Table Space Container Utilization ” on page 539
- “ts.ts_util - Table Space Utilization ” on page 538

tsc.tscont_op_status - Table Space Container Operational State

Identifier	tsc.tscont_op_status
Health monitor level	Table Space Container
Category	Table Space Storage
Type	State-based
Unit	Not applicable

Description

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.

Related reference:

- “container_accessible - Accessibility of Container ” on page 346
- “Health indicators” on page 503
- “tablespace_num_containers - Number of Containers in Table Space ” on page 343
- “tsc.tscont_util - Table Space Container Utilization ” on page 539
- “ts.ts_op_status - Table Space Operational State ” on page 540
- “ts.ts_util - Table Space Utilization ” on page 538

Sorting health indicators

db2.sort_privmem_util - Private Sort Memory Utilization

Identifier	db2.sort_privmem_util
Health monitor level	Database
Category	Sorting
Type	Upper-bounded threshold-based
Unit	Percentage

Description

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily.

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.

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-water mark 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`.

Related reference:

- “db.spilled_sorts - Percentage of Sorts That Overflowed ” on page 543
- “Health indicators” on page 503
- “db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization ” on page 544
- “db.sort_shrmem_util - Shared Sort Memory Utilization ” on page 542

db.sort_shrmem_util - Shared Sort Memory Utilization

Identifier	db.sort_shrmem_util
Health monitor level	Database
Category	Sorting
Type	Upper-bounded threshold-based
Unit	Percentage

Description

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily.

This indicator tracks the utilization of the shared sort memory. The `sheapthres_shr` database configuration parameter is a hard limit. If the allocation is close to the limit, an alert may be generated.

The indicator is calculated using the formula:

$$(db.sort_shrheap_allocated / sheapthres_shr) * 100$$

Note that if `sheapthres_shr` is set to 0, then `sheapthres` serves as the shared sortheap threshold.

The Maximum Shared Sort Memory Used snapshot monitor element maintains a shared sort memory high-water mark 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.

Related reference:

- “Health indicators” on page 503
- “db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization ” on page 544
- “db.spilled_sorts - Percentage of Sorts That Overflowed ” on page 543
- “db2.sort_privmem_util - Private Sort Memory Utilization ” on page 541

db.spilled_sorts - Percentage of Sorts That Overflowed

Identifier	db.spilled_sorts
Health monitor level	Database
Category	Sorting
Type	Upper-bounded threshold-based
Unit	Percentage

Description

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily.

Sorts that overflow to disk can cause significant performance degradation. If this occurs, an alert may be generated.

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.

Related concepts:

- “Self tuning memory” in *Performance Guide*

Related reference:

Health indicators

- “db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization ” on page 544
- “db.sort_shrmem_util - Shared Sort Memory Utilization ” on page 542
- “db2.sort_privmem_util - Private Sort Memory Utilization ” on page 541
- “Health indicators” on page 503
- “sort_overflows - Sort Overflows ” on page 211
- “total_sorts - Total Sorts ” on page 210

db.max_sort_shrmem_util - Long Term Shared Sort Memory Utilization

Identifier	db.max_sort_shrmem_util
Health monitor level	Database
Category	Sorting
Type	Lower-bounded threshold-based
Unit	Percentage

Description

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily.

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 DB2.

An alert might be generated when the percentage usage is low.

The indicator is calculated using the formula:

$$(db.max_shr_sort_mem / sheaphres_shr) * 100$$

The system monitor element db.max_shr_sort_mem (based on the sort_shrheap_top monitor element) is the high-water mark 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.

Related concepts:

- “Self tuning memory” in *Performance Guide*

Related reference:

- “Health indicators” on page 503
- “db.spilled_sorts - Percentage of Sorts That Overflowed ” on page 543
- “db2.sort_privmem_util - Private Sort Memory Utilization ” on page 541
- “db.sort_shrmem_util - Shared Sort Memory Utilization ” on page 542

Database manager (DBMS) health indicators

db2.db2_op_status - Instance Operational State

Identifier	db2.db2_op_status
-------------------	-------------------

Health monitor level	Instance
Category	DBMS
Type	State-based
Unit	Not applicable

Description

An instance is considered healthy if the instance state does not restrict activity or tasks being performed.

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.

Related reference:

- “db2_status - Status of DB2 Instance ” on page 152
- “Health indicators” on page 503
- “Instance Highest Severity Alert State ” on page 545

Instance Highest Severity Alert State

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

Description

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. The order of the alert states is as follows:

- Alarm
- Warning
- Attention
- Normal

The alert state of the instance determines the overall health of DB2.

Related reference:

- “Health indicators” on page 503
- “db2.db2_op_status - Instance Operational State ” on page 544

Database health indicators

db.db_op_status - Database Operational State

Identifier	db.db_op_status
Health monitor level	Database
Category	Database
Type	State-based
Unit	Not applicable

Description

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.

Related reference:

- “db_status - Status of Database ” on page 156
- “Database Highest Severity Alert State ” on page 546
- “Health indicators” on page 503

Database Highest Severity Alert State

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

Description

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. The order of the alert states is as follows:

- Alarm
- Warning
- Attention
- Normal

Related reference:

- “db.db_op_status - Database Operational State ” on page 546
- “Health indicators” on page 503

Maintenance health indicators

db.tb_reorg_req - Reorganization Required

Identifier	db.tb_reorg_req
Health monitor level	Database

Category	Database Maintenance
Type	Collection state-based
Unit	Not applicable

Description:

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 could yield an improved performance and freed space in the table or indexes .

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.

Related reference:

- “REORG INDEXES/TABLE command” in *Command Reference*
- “Health indicators” on page 503

db.tb_runstats_req - Statistics Collection Required

Identifier	db.tb_runstats_req
Health monitor level	Database
Category	Database Maintenance
Type	Collection state-based
Unit	Not applicable

Description

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.

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 could not complete successfully.

Related reference:

Health indicators

- “Health indicators” on page 503
- “RUNSTATS command” in *Command Reference*

db.db_backup_req - Database Backup Required

Identifier	db.db_backup_req
Health monitor level	Database
Category	Database Maintenance
Type	State-based
Unit	Not applicable

Description

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.

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 could not complete successfully.

Related reference:

- “last_backup - Last Backup Timestamp ” on page 158

High availability disaster recovery health indicators

db.hadr_op_status - HADR Operational Status

Identifier	db.hadr_op_status
Health monitor level	Database
Category	High availability disaster recovery
Type	State-based
Unit	Not applicable

Description

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 could generate an Attention alert.

Related concepts:

- “High availability disaster recovery overview” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “hadr_connect_status - HADR Connection Status monitor element” on page 429

- “hadr_state - HADR State monitor element” on page 428
- “db.hadr_delay - HADR Log Delay ” on page 549
- “Health indicators” on page 503

db.hadr_delay - HADR Log Delay

Identifier	db.hadr_delay
Health monitor level	Database
Category	High availability disaster recovery
Type	Upper-bounded threshold-based
Unit	Minutes

Description

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 it is possible for data loss to occur when failing over to the standby database after a failure on the primary database. A large delay value could also mean longer downtime when takeover is required since the primary is ahead of the standby.

Related concepts:

- “High availability disaster recovery overview” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “hadr_syncmode - HADR synchronization mode for log write in peer state configuration parameter” in *Performance Guide*
- “db.hadr_op_status - HADR Operational Status ” on page 548
- “Health indicators” on page 503

Logging health indicators

db.log_util - Log Utilization

Identifier	db.log_util
Health monitor level	Database
Category	Logging
Type	Upper-bounded threshold-based
Unit	Percentage

Description

This indicator tracks the total amount of active log space used in bytes in the database.

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:

$$(db.total_log_used / (db.total_log_used + db.total_log_available)) * 100$$

Health indicators

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.

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “Health indicators” on page 503
- “db.log_fs_util - Log Filesystem Utilization” on page 550

db.log_fs_util - Log Filesystem Utilization

Identifier	db.log_fs_util
Health monitor level	Database
Category	Logging
Type	Upper-bounded threshold-based
Unit	Percentage

Description

Log Filesystem Utilization tracks the fullness of the file system on which the transaction logs reside. DB2 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 (i.e. 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 userexit 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.

Related reference:

- “Health indicators” on page 503
- “db.log_util - Log Utilization” on page 549

Application concurrency health indicators

db.deadlock_rate - Deadlock Rate

Identifier	db.deadlock_rate
Health monitor level	Database
Category	Application Concurrency

Type	Upper-bounded threshold-based
Unit	Deadlocks per hour

Description

Deadlock rate tracks the rate at which deadlocks are occurring in the database and the degree to which applications are experiencing contention problems. 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.

Related reference:

- "Health indicators" on page 503
- "db.lock_escal_rate - Lock Escalation Rate" on page 552
- "db.locklist_util - Lock List Utilization " on page 551
- "db.apps_waiting_locks - Percentage of Applications Waiting on Locks" on page 553

db.locklist_util - Lock List Utilization

Identifier	db.locklist_util
Health monitor level	Database
Category	Application Concurrency
Type	Upper-bounded threshold-based
Unit	Percentage

Description

This indicator tracks the amount of lock list memory that is being used. 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.

Health indicators

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.

Related concepts:

- “Self tuning memory” in *Performance Guide*

Related reference:

- “db.deadlock_rate - Deadlock Rate” on page 550
- “Health indicators” on page 503
- “db.lock_escal_rate - Lock Escalation Rate” on page 552
- “db.apps_waiting_locks - Percentage of Applications Waiting on Locks” on page 553

db.lock_escal_rate - Lock Escalation Rate

Identifier	db.lock_escal_rate
Health monitor level	Database
Category	Application Concurrency
Type	Upper-bounded threshold-based
Unit	Lock escalations per hour

Description

This indicator tracks the rate at which locks have been escalated from row locks to a table lock thereby impacting transaction concurrency.

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:

$$(db.lock_escals_t - db.lock_escals_{t-1})$$

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.

Related concepts:

- “Self tuning memory” in *Performance Guide*

Related reference:

- “db.deadlock_rate - Deadlock Rate” on page 550
- “Health indicators” on page 503
- “db.locklist_util - Lock List Utilization ” on page 551
- “db.apps_waiting_locks - Percentage of Applications Waiting on Locks” on page 553

db.apps_waiting_locks - Percentage of Applications Waiting on Locks

Identifier	db.apps_waiting_locks
Health monitor level	Database
Category	Application Concurrency
Type	Upper-bounded threshold-based
Unit	Percentage

Description

This indicator measures the percentage of all currently executing applications that are waiting on locks.

A high percentage can indicate that applications are experiencing concurrency problems which can negatively affect performance.

The indicator is calculated using the formula:

$$(db.locks_waiting / db.apps_cur_cons) * 100$$

Related reference:

- “db.deadlock_rate - Deadlock Rate” on page 550
- “Health indicators” on page 503
- “db.lock_escal_rate - Lock Escalation Rate” on page 552
- “db.locklist_util - Lock List Utilization ” on page 551

Package and catalog caches, and workspaces health indicators

db.catcache_hitratio - Catalog Cache Hit Ratio

Identifier	db.catcache_hitratio
Health monitor level	Database
Category	Package and Catalog Caches, and Workspaces

Health indicators

Type Lower-bounded threshold-based

Unit Percentage

Description

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.

The indicator is calculated using the formula:

$$(1 - (\text{db.cat_cache_inserts} / \text{db.cat_cache_lookups})) * 100$$

Related reference:

- “Health indicators” on page 503
- “db.pkgcache_hitratio - Package Cache Hit Ratio ” on page 554
- “db.shrworkspace_hitratio - Shared Workspace Hit Ratio ” on page 554

db.pkgcache_hitratio - Package Cache Hit Ratio

Identifier db.pkgcache_hitratio

Health monitor level Database

Category Package and Catalog Caches, and Workspaces

Type Lower-bounded threshold-based

Unit Percentage

Description

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.

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.

Related concepts:

- “Self tuning memory” in *Performance Guide*

Related reference:

- “db.catcache_hitratio - Catalog Cache Hit Ratio” on page 553
- “Health indicators” on page 503
- “db.shrworkspace_hitratio - Shared Workspace Hit Ratio ” on page 554

db.shrworkspace_hitratio - Shared Workspace Hit Ratio

Identifier db.shrworkspace_hitratio

Health monitor level Database

Category Package and Catalog Caches, and Workspaces

Type Lower-bounded threshold-based

Unit Percentage

Description

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.

The indicator is calculated using the formula:

$$(1 - (\text{db.shr_workspace_section_inserts} / \text{db.shr_workspace_section_lookups})) * 100$$

Related reference:

- “db.catcache_hitratio - Catalog Cache Hit Ratio” on page 553
- “Health indicators” on page 503
- “db.pkgcache_hitratio - Package Cache Hit Ratio ” on page 554

Memory health indicators

db2.mon_heap_util - Monitor Heap Utilization

Identifier db2.mon_heap_util

Health monitor level Instance

Category Memory

Type Upper-bounded threshold-based

Unit Percentage

Description

This indicator tracks the consumption of the monitor heap memory, based on the memory pool with the ID SQLM_HEAP_MONITOR.

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.

Related reference:

- “db.db_heap_util - Database Heap Utilization ” on page 555
- “Health indicators” on page 503

db.db_heap_util - Database Heap Utilization

Identifier db.db_heap_util

Health monitor level Database

Category Memory

Type Upper-bounded threshold-based

Unit Percentage

Health indicators

Description

This indicator tracks the consumption of the monitor heap memory, based on the memory pool with the ID `SQLM_HEAP_DATABASE`.

The utilization is calculated using the formula

$$(db.pool_cur_size / db.pool_max_size) * 100$$

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.

Related reference:

- “Health indicators” on page 503
- “db2.mon_heap_util - Monitor Heap Utilization ” on page 555

Federated health indicators

db.fed_nicknames_op_status - Nickname Status

Identifier	db.fed_nicknames_op_status
Health monitor level	Database
Category	Federated
Type	Collection state-based
Unit	Not applicable

Description:

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.

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.

Related reference:

- “ALTER NICKNAME statement” in *SQL Reference, Volume 2*
- “CREATE NICKNAME statement” in *SQL Reference, Volume 2*
- “CREATE USER MAPPING statement” in *SQL Reference, Volume 2*
- “DROP statement” in *SQL Reference, Volume 2*

db.fed_servers_op_status - Data Source Server Status

Identifier	db.fed_servers_op_status
Health monitor level	Database
Category	Federated
Type	Collection state-based

Unit Not applicable

Description:

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.

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 data source server status.

Related reference:

- “ALTER SERVER statement” in *SQL Reference, Volume 2*
- “ALTER USER MAPPING statement” in *SQL Reference, Volume 2*
- “CREATE USER MAPPING statement” in *SQL Reference, Volume 2*

Health indicators

Chapter 12. Health Monitor Interfaces

Health monitor interfaces

The following table lists the health monitor interfaces for APIs:

Table 833. 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 834. 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:

Health Monitor Interfaces

Table 835. Health monitor interfaces: SQL functions

Monitoring task	SQL Function
Database manager level health information snapshot	HEALTH_DBM_INFO
Database manager level health indicator snapshot	HEALTH_DBM_HI
Database manager level health indicator history snapshot	HEALTH_DBM_HI_HIS
Database level health information snapshot	HEALTH_DB_INFO
Database level health indicator snapshot	HEALTH_DB_HI
Database level health indicator history snapshot	HEALTH_DB_HI_HIS
Database level health indicator collection snapshot	HEALTH_DB_HIC
Database level health indicator collection history snapshot	HEALTH_DB_HIC_HIS
Table space level health information snapshot	HEALTH_TBS_INFO
Table space level health indicator snapshot	HEALTH_TBS_HI
Table space level health indicator history snapshot	HEALTH_TBS_HI_HIS
Table space container level health information snapshot	HEALTH_CONT_INFO
Table space container level health indicator snapshot	HEALTH_CONT_HI
Table space container level health indicator history snapshot	HEALTH_CONT_HI_HIS

Related reference:

- “db2ConvMonStream API - Convert the monitor stream to the pre-version 6 format” in *Administrative API Reference*
- “db2GetSnapshot API - Get a snapshot of the database manager operational status” in *Administrative API Reference*
- “db2GetSnapshotSize API - Estimate the output buffer size required for the db2GetSnapshot API” in *Administrative API Reference*
- “GET HEALTH SNAPSHOT command” in *Command Reference*

Part 5. Appendixes

Appendix A. DB2 Database technical information

Overview of the DB2 technical information

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
 - Topics
 - Help for DB2 tools
 - Sample programs
 - Tutorials
- DB2 books
 - PDF files (downloadable)
 - PDF files (from the DB2 PDF CD)
 - printed books
- Command line help
 - Command help
 - Message help
- Sample programs

IBM periodically makes documentation updates available. If you access the online version on the DB2 Information Center at ibm.com[®], you do not need to install documentation updates because this version is kept up-to-date by IBM. If you have installed the DB2 Information Center, it is recommended that you install the documentation updates. Documentation updates allow you to update the information that you installed from the *DB2 Information Center CD* or downloaded from Passport Advantage as new information becomes available.

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 Redbooks™ online at 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 we can improve the DB2 documentation, send an e-mail 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 e-mail 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.

Related concepts:

- “Features of the DB2 Information Center” in *Online DB2 Information Center*
- “Sample files” in *Samples Topics*

Related tasks:

- “Invoking command help from the command line processor” in *Command Reference*
- “Invoking message help from the command line processor” in *Command Reference*
- “Updating the DB2 Information Center installed on your computer or intranet server” on page 569

Related reference:

- “DB2 technical library in hardcopy or PDF format” on page 564

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. DB2 Version 9 manuals in PDF format can be downloaded from www.ibm.com/software/data/db2/udb/support/manualsv9.html.

Although the tables identify books available in print, the books might not be available in your country or region.

The information in these books is fundamental to all DB2 users; you will find this information useful whether you are a programmer, a database administrator, or someone who works with DB2 Connect or other DB2 products.

Table 836. DB2 technical information

Name	Form Number	Available in print
<i>Administration Guide: Implementation</i>	SC10-4221	Yes
<i>Administration Guide: Planning</i>	SC10-4223	Yes
<i>Administrative API Reference</i>	SC10-4231	Yes
<i>Administrative SQL Routines and Views</i>	SC10-4293	No
<i>Call Level Interface Guide and Reference, Volume 1</i>	SC10-4224	Yes
<i>Call Level Interface Guide and Reference, Volume 2</i>	SC10-4225	Yes
<i>Command Reference</i>	SC10-4226	No
<i>Data Movement Utilities Guide and Reference</i>	SC10-4227	Yes
<i>Data Recovery and High Availability Guide and Reference</i>	SC10-4228	Yes
<i>Developing ADO.NET and OLE DB Applications</i>	SC10-4230	Yes
<i>Developing Embedded SQL Applications</i>	SC10-4232	Yes

Table 836. DB2 technical information (continued)

Name	Form Number	Available in print
<i>Developing SQL and External Routines</i>	SC10-4373	No
<i>Developing Java Applications</i>	SC10-4233	Yes
<i>Developing Perl and PHP Applications</i>	SC10-4234	No
<i>Getting Started with Database Application Development</i>	SC10-4252	Yes
<i>Getting started with DB2 installation and administration on Linux and Windows</i>	GC10-4247	Yes
<i>Message Reference Volume 1</i>	SC10-4238	No
<i>Message Reference Volume 2</i>	SC10-4239	No
<i>Migration Guide</i>	GC10-4237	Yes
<i>Net Search Extender Administration and User's Guide</i> Note: HTML for this document is not installed from the HTML documentation CD.	SH12-6842	Yes
<i>Performance Guide</i>	SC10-4222	Yes
<i>Query Patroller Administration and User's Guide</i>	GC10-4241	Yes
<i>Quick Beginnings for DB2 Clients</i>	GC10-4242	No
<i>Quick Beginnings for DB2 Servers</i>	GC10-4246	Yes
<i>Spatial Extender and Geodetic Data Management Feature User's Guide and Reference</i>	SC18-9749	Yes
<i>SQL Guide</i>	SC10-4248	Yes
<i>SQL Reference, Volume 1</i>	SC10-4249	Yes
<i>SQL Reference, Volume 2</i>	SC10-4250	Yes
<i>System Monitor Guide and Reference</i>	SC10-4251	Yes
<i>Troubleshooting Guide</i>	GC10-4240	No
<i>Visual Explain Tutorial</i>	SC10-4319	No
<i>What's New</i>	SC10-4253	Yes
<i>XML Extender Administration and Programming</i>	SC18-9750	Yes
<i>XML Guide</i>	SC10-4254	Yes
<i>XQuery Reference</i>	SC18-9796	Yes

Table 837. DB2 Connect-specific technical information

Name	Form Number	Available in print
<i>DB2 Connect User's Guide</i>	SC10-4229	Yes

Table 837. DB2 Connect-specific technical information (continued)

Name	Form Number	Available in print
Quick Beginnings for DB2 Connect Personal Edition	GC10-4244	Yes
Quick Beginnings for DB2 Connect Servers	GC10-4243	Yes

Table 838. WebSphere® Information Integration technical information

Name	Form Number	Available in print
WebSphere Information Integration: Administration Guide for Federated Systems	SC19-1020	Yes
WebSphere Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018	Yes
WebSphere Information Integration: Configuration Guide for Federated Data Sources	SC19-1034	No
WebSphere Information Integration: SQL Replication Guide and Reference	SC19-1030	Yes

Note: The DB2 Release Notes provide additional information specific to your product's release and fix pack level. For more information, see the related links.

Related concepts:

- "Overview of the DB2 technical information" on page 563
- "About the Release Notes" in *Release notes*

Related tasks:

- "Ordering printed DB2 books" on page 566

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 CD* 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 CD 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 CD 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/db2help/>.

Procedure:

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:
 - 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.
 - When you call, specify that you want to order a DB2 publication.
 - Provide your representative with the titles and form numbers of the books that you want to order.

Related concepts:

- "Overview of the DB2 technical information" on page 563

Related reference:

- "DB2 technical library in hardcopy or PDF format" on page 564

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.

Procedure:

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.

Related tasks:

- "Invoking command help from the command line processor" in *Command Reference*
- "Invoking message help from the command line processor" in *Command Reference*

Accessing different versions of the DB2 Information Center

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/>.

Related tasks:

- “Updating the DB2 Information Center installed on your computer or intranet server” on page 569

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.

Procedure:

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 **Tools** → **Options** → **Languages** button. 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.

Related concepts:

- “Overview of the DB2 technical information” on page 563

Updating the DB2 Information Center installed on your computer or intranet server

If you have a locally-installed DB2 Information Center, updated topics can be available for download. The 'Last updated' value found at the bottom of most topics indicates the current level for that topic.

To determine if there is an update available for the entire DB2 Information Center, look for the 'Last updated' value on the Information Center home page. Compare the value in your locally installed home page to the date of the most recent downloadable update at <http://www.ibm.com/software/data/db2/udb/support/icupdate.html>. You can then update your locally-installed Information Center if a more recent downloadable update is available.

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 download and apply updates.
2. Use the Update feature to determine if update packages are available from IBM.

Note: Updates are also available on CD. For details on how to configure your Information Center to install updates from CD, see the related links. If update packages are available, use the Update feature to download the packages. (The Update feature is only available in stand-alone mode.)

3. Stop the stand-alone Information Center, and restart the DB2 Information Center service on your computer.

Procedure:

To update the DB2 Information Center installed on your computer or intranet server:

1. Stop the DB2 Information Center service.
 - 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/db2icdv9 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 C:\Program Files\IBM\DB2 Information Center\Version 9 directory.
 - c. Run the help_start.bat file using the fully qualified path for the DB2 Information Center:
`<DB2 Information Center dir>\doc\bin\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 directory.
- b. Run the help_start script using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>/doc/bin/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 download process, check the selections you want to download, then click **Install Updates**.
5. After the download and installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center.

- On Windows, run the help_end.bat file using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>\doc\bin\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, run the help_end script using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>/doc/bin/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 service.
 - 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/db2icdv9 start
```

The updated DB2 Information Center displays the new and updated topics.

Related concepts:

- “DB2 Information Center installation options” in *Quick Beginnings for DB2 Servers*

Related tasks:

- “Installing the DB2 Information Center using the DB2 Setup wizard (Linux)” in *Quick Beginnings for DB2 Servers*
- “Installing the DB2 Information Center using the DB2 Setup wizard (Windows)” in *Quick Beginnings for DB2 Servers*

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.

Native XML data store

Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

Visual Explain Tutorial

Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

Related concepts:

- “Visual Explain overview” in *Administration Guide: Implementation*

DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 products.

DB2 documentation

Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

DB2 Technical Support Web site

Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs or bug fixes), fix packs, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at <http://www.ibm.com/software/data/db2/udb/support.html>

Related concepts:

- “Introduction to problem determination” in *Troubleshooting Guide*
- “Overview of the DB2 technical information” on page 563

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

IBM may not offer the products, services, or features discussed in this document in all 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

Company, product, or service names identified in the documents of the DB2 Version 9 documentation library may be trademarks or service marks of International Business Machines Corporation or other companies. Information on the trademarks of IBM Corporation in the United States, other countries, or both is located at <http://www.ibm.com/legal/copytrade.shtml>.

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 documentation library:

Microsoft[®], Windows, Windows NT[®], and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Itanium[®], Pentium[®], and Xeon[®] are trademarks of Intel Corporation in the United States, other countries, or both.

Java[™] and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Index

A

acc_curs_blk element 371
accepted block cursor requests monitor element 371
accesses to overflowed records monitor element 357
active sorts monitor element 212
active_hash_joins monitor element 215
active_sorts element 212
Activity Monitor 488, 492
agent ID holding lock monitor element 321
agent_id element 163
agent_id_holding_lock element 321
agent_pid element 189
agent_status element 446
agent_sys_cpu_time element 413
agent_usr_cpu_time element 413
agents and connections
 associated agent
 stolen agent 190
 coordinator agent 190
 primed agent 190
 subagent 190
agents assigned from pool monitor element 198
agents created due to empty agent pool monitor element 198
agents registered monitor element 196
agents waiting for a token monitor element 196
agents_created_empty_pool element 198
agents_from_pool element 198
agents_registered element 196
agents_registered_top element 196
agents_stolen element 199
agents_top element 412
agents_waiting_on_token element 196
agents_waiting_top element 197
alerts
 enabling 507
amount of free space available monitor element 159
amount of log space accounted for by dirty pages monitor element 294
amount of log to be redone for recovery monitor element 295
amount of space used on a file system monitor element 160
API request types
 health monitor 519
appl_con_time element 183
appl_id element 169
appl_id_holding_lk element 322
appl_id_oldest_xact element 167
appl_idle_time element 187
appl_name element 168
appl_priority element 179
appl_priority_type element 180
appl_section_inserts element 281

appl_section_lookups element 280
appl_status element 164
application agent priority monitor element 179
application creator monitor element 390
application handle (agent ID) monitor element 163
application ID holding lock monitor element 322
application ID monitor element 169
application idle time monitor element 187
application name monitor element 168
application priority type monitor element 180
application status change time monitor element 167
application status monitor element 164
application with oldest transaction monitor element 167
applications currently connected monitor element 194
applications currently executing in database monitor element 195
appls_cur_cons element 194
appls_in_db2 element 195
associated_agents_top element 200
auth_id element 172
authority_lvl element 180
authorization ID monitor element 172
automatic storage path monitor element 159

B

binds_precompiles element 383
binds/precompiles attempted monitor element 383
blocked event monitors 74
blocking cursor monitor element 471
blocking_cursor element 471
bp_cur_buffsz monitor element 267
bp_id monitor element 232
bp_name element 255
bp_new_buffsz monitor element 268
bp_pages_left_to_remove monitor element 268
bp_tbsp_use_count monitor element 268
buff_free element 219
buff_free_bottom element 219
buffer pool activity
 monitor elements 229
buffer pool asynchronous data reads monitor element 245
buffer pool asynchronous data writes monitor element 246
buffer pool asynchronous index read requests monitor element 251
buffer pool asynchronous index reads monitor element 248
buffer pool asynchronous index writes monitor element 247
buffer pool asynchronous read requests monitor element 250
buffer pool asynchronous read time monitor element 249
buffer pool asynchronous write time monitor element 249
buffer pool asynchronous xda data reads monitor element 260
buffer pool asynchronous xda data writes monitor element 261
buffer pool asynchronous xda read requests monitor element 259
buffer pool currently being used monitor element 332
buffer pool data logical reads monitor element 233
buffer pool data physical reads monitor element 235
buffer pool data writes monitor element 237
buffer pool ID monitor element 232
buffer pool index logical reads monitor element 238
buffer pool index physical reads monitor element 240
buffer pool index writes monitor element 241
buffer pool log space cleaners triggered monitor element 251
buffer pool no victim buffers monitor element 254
buffer pool temporary data logical reads monitor element 234
buffer pool temporary data physical reads monitor element 236
buffer pool temporary index logical reads monitor element 239
buffer pool temporary index physical reads monitor element 241
buffer pool temporary xda data logical reads monitor element 265
buffer pool temporary xda data physical reads monitor element 266
buffer pool that will be used at next startup monitor element 332
buffer pool threshold cleaners triggered monitor element 254
buffer pool victim page cleaners triggered monitor element 252
buffer pool xda data logical reads monitor element 262
buffer pool xda data physical reads monitor element 263
buffer pool xda data writes monitor element 264
bufferpool name monitor element 255
byte order of event data monitor element 422
byte_order element 422

C

capturing health snapshots
 using a client application 516
 using CLP 515
 using SQL 513

cat_cache_inserts element 274

cat_cache_lookups element 273

cat_cache_overflows element 275

cat_cache_size_top element 276

catalog cache high water mark monitor element 276

catalog cache hit ratio health indicator 553

catalog cache inserts monitor element 274

catalog cache lookups monitor element 273

catalog cache overflows monitor element 275

catalog node monitor elements
 catalog node network name monitor element 157
 catalog node number monitor element 158

catalog_node element 158

catalog_node_name element 157

ch_free monitor element 220

ch_free_bottom monitor element 220

channels currently free monitor element 220

client application
 capturing health snapshots 516

client communication protocol monitor element 178

client operating platform monitor element 178

client process ID monitor element 177

client product and version ID monitor element 174

client_db_alias element 174

client_nname element 173

client_pid element 177

client_platform element 178

client_prdid element 174

client_protocol element 178

CLP
 capturing health snapshots 515

CLP commands
 for health monitor 516

codepage_id element 166

collection state-based health indicators 503

comm_private_mem element 200

commit statements attempted monitor element 375

commit_sql_stmts element 375

committed private memory monitor element 200

communication error time monitor element 471

communication errors monitor element 471

comp_env_desc element 401

compilation environment handle monitor element 401

completed progress work units monitor element 227

con_elapsed_time monitor element 470

con_local_databases monitor element 193

con_response_time monitor element 470

configuration NNAME at monitoring (server) node monitor element 149

configuration NNAME of client monitor element 173

conn_complete_time monitor element 184

conn_time monitor element 156

connection request start timestamp monitor element 183

connection status monitor element 221

connection switches monitor element 202

connection_status element 221

connections involved in deadlock monitor element 311

connections_top element 183

connects since database activation monitor element 194

consistency_token monitor element 388

contacting IBM 579

container accessibility monitor element 346

container identification monitor element 344

container name monitor element 344

container type monitor element 344

container_accessible monitor element 346

container_id monitor element 344

container_name monitor element 344

container_stripe_set monitor element 345

container_total_pages monitor element 345

container_type monitor element 344

container_usable_pages monitor element 345

control table message monitor element 426

coord_agent_pid monitor element 189

coord_agents_top monitor element 199

coord_node monitor element 182

coordinating node monitor element 182

coordinator agent monitor element 189

corr_token monitor element 177

count monitor element 421

counters, data element type 7

country_code monitor element, renamed to database territory code monitor element 179

create nickname response time monitor element 481

create nicknames monitor element 477

create_nickname element 477

create_nickname_time element 481

creator monitor element 390

current active log file number monitor element 300

current archive log file number monitor element 300

current number of connections for DB2 Connect monitor element 442

current number of tablequeue buffers overflowed monitor element 407

current page being processed in table reorganize monitor element 366

current progress list attributes monitor element 225

current progress list sequence number monitor element 225

current rebalancer priority monitor element 340

current size of buffer pool monitor element 267

current size of storage pool monitor element 204

current table space size monitor element 335

current_active_log element 300

current_archive_log element 300

cursor name monitor element 389

cursor_name monitor element 389

D

data element types
 counters 7
 description 3

data object pages monitor element 361

data partition identifier monitor element 188

data source name monitor element 475

data source server status health indicator 556

data_object_pages element 361

data_partition_id 188

database activation timestamp monitor element 155

database alias at the gateway monitor element 440

database alias used by application monitor element 174

database backup required health indicator 548

database configuration
 monitor elements 229

database connections
 applications connected currently, monitor element 194
 applications executing in the database currently, monitor element 195
 connection request completion timestamp, monitor element 184

database deactivation timestamp monitor element 156

database files closed monitor element 244

database heap utilization health indicator 555

database highest severity alert state health indicator 546

database location monitor element 157

database manager configuration
 monitor elements 189

database manager type at monitored (server) node monitor element 150

database monitor
 description 3

database name monitor element 154

database operated on by utility monitor element 223

- database operational state
 - health indicators 546
- database path monitor element 155
- database system events
 - collecting monitor information 61
- database system monitor
 - data organization 3
 - description 3
 - interfaces 485
 - memory requirements 9
 - output 8
 - restricting collection of monitor data 13
 - sample 485
 - self-describing data stream 8
- Database Territory Code monitor element 179
- datasource_name element 475
- db_conn_time element 155
- db_heap_top element 288
- db_location element 157
- db_name element 154
- db_path element 155
- db_status element 156
- db_storage_path element 159
- db.alert_state
 - health indicator 546
- db.apps_waiting_locks health indicator 553
- db.catcache_hitratio health indicator 553
- db.database_heap_utilization health indicator 555
- db.db_auto_storage_util 536
- db.db_backup_req health indicator 548
- db.db_op_status health indicators 546
- db.deadlock_rate health indicator 550
- db.fed_nicknames_status health indicator 556
- db.fed_servers_status health indicator 556
- db.hadr_delay health indicator 549
- db.hadr_op_status health indicator 548
- db.lock_escal_rate health indicator 552
- db.locklist_utilization health indicator 551
- db.log_fs_utilization health indicator 550
- db.log_utilization health indicator 549
- db.max_sort_shrmem_util health indicator 544
- db.pkgcache_hitratio health indicator 554
- db.shrworkspace_hitratio health indicator 554
- db.sort_shrmem_util health indicator 542
- db.spilled_sorts health indicator 543
- db.tb_reorg_req health indicator 546
- db.tb_runstats_req health indicator 547
- DB2 connect
 - monitor elements 437
- DB2 Connect gateway first connect initiated monitor element 441
- DB2 Information Center
 - updating 569
 - versions 568
 - viewing in different languages 568

- db2_status element 152
- db2.db2_alert_state health indicator 545
- db2.db2_op_status health indicator 544
- db2.mon_heap_utilization health indicator 555
- db2.sort_privmem_util health indicator 541
- db2event.ctl 73
- db2start_time element 148
- DBMS highest severity alert state health indicator 545
- DCS application agents monitor element 446
- DCS application status monitor element 446
- DCS database name monitor element 440
- dcs_appl_status element 446
- dcs_db_name element 440
- ddl_sql_stmts element 378
- deadlock event identifier monitor element 312
- deadlock rate health indicator 550
- deadlock_id element 312
- deadlock_node element 312
- deadlocks detected monitor element 303
- deadlocks element 303
- degree of parallelism monitor element 412
- degree_parallelism element - 412
- delete response time monitor element 481
- delete_sql_stmts element 476
- delete_time element 481
- deletes monitor element 476
- direct read requests monitor element 270
- direct read time monitor element 272
- direct reads from database monitor element 269
- direct write requests monitor element 271
- direct write time monitor element 272
- direct writes to database monitor element 269
- direct_read_reqs element 270
- direct_read_time element 272
- direct_reads element 269
- direct_write_reqs element 271
- direct_write_time element 272
- direct_writes element 269
- disconn_time element 156
- disconnects element 475
- disconnects monitor element 475
- dl_conns element 311
- DMS table spaces
 - health indicators 535
- documentation 563, 564
 - terms and conditions of use 572
- drda correlation token monitor element 177
- dynamic SQL
 - monitor elements 409
- dynamic SQL statements attempted monitor element 373
- dynamic_sql_stmts element 373

E

- elapsed statement execution time monitor element 411
- elapsed time spent on DB2 Connect gateway processing monitor element 443
- elapsed_exec_time element 467
- enabling health alerts 507
- end stripe monitor element 348
- event monitor name monitor element 423
- event monitoring
 - monitor elements 421
- event monitors
 - blocked 74
 - buffers 74
 - creating
 - event monitor 63
 - file event monitor 71
 - pipe event monitor 75
 - table event monitor 64
 - database system events 61
 - definition 59
 - event records 90
 - file management 73
 - formatting output from command line 79
 - named pipe management 76
 - non-blocked 74
 - on partitioned databases 77
 - output, self-describing data stream 91
 - partitioned databases 77
 - table management 67
 - transferring event data between systems 93
 - type mappings to logical data groups 129
- event records, finding corresponding applications 90
- event start time monitor element 392
- event stop time monitor element 391
- event time monitor element 424
- event type mappings to logical data groups 129
- event_monitor_name element 423
- event_time element 424
- evmon_activates element 425
- evmon_flushes element 424
- exclusive lock escalations monitor element 305
- execution_id element 176

F

- failed statement operations monitor element 374
- failed_sql_stmts element 374
- fast communications manager monitor elements 219
- fcm buffers currently free monitor element 219
- federated database systems
 - monitor elements 474
- fetch_count element 394

- file event monitors
 - buffering 74
 - creating 71
 - file management 73
 - formatting output from command line 79
- file system caching monitor element 350
- File system type monitor element 162
- files_closed element 244
- first active log file number monitor element 299
- first_active_log element 299
- first_overflow_time element 422
- format
 - health indicator 533
- free pages in table space monitor element 334
- fs_caching element 350
- fs_free_size element 159
- fs_id element 161
- fs_total_size element 160
- fs_type element 162
- fs_used_size element 160

G

- global health snapshots 522
- global snapshots on partitioned database systems 53
- graphical tools, health monitor 523
- gw_comm_error_time element 471
- gw_comm_errors element 471
- gw_con_time element 441
- gw_connections_top element 441
- gw_cons_wait_client element 443
- gw_cons_wait_host element 442
- gw_cur_cons element 442
- gw_db_alias element 440
- gw_exec_time element 443
- gw_total_cons element 441

H

- HADR connection status monitor element 429
- HADR connection time monitor element 430
- HADR heartbeat monitor element 430
- HADR local host monitor element 431
- HADR local service monitor element 432
- HADR log delay health indicator 549
- HADR log gap monitor element 437
- HADR operational status health indicator 548
- HADR primary log file monitor element 434
- HADR primary log lsn monitor element 435
- HADR primary log page monitor element 434
- HADR remote host monitor element 432
- HADR remote instance monitor element 433
- HADR remote service monitor element 433

- HADR role monitor element 427
- HADR standby log file monitor element 435
- HADR standby log lsn monitor element 436
- HADR standby log page monitor element 436
- HADR state monitor element 428
- HADR synchronization mode monitor element 428
- HADR timeout monitor element 433
- hadr_connect_status element 429
- hadr_connect_time element 430
- hadr_heartbeat element 430
- hadr_local_host element 431
- hadr_local_service element 432
- hadr_log_gap element 437
- hadr_primary_log_file element 434
- hadr_primary_log_lsn element 435
- hadr_primary_log_page element 434
- hadr_remote_host element 432
- hadr_remote_instance element 433
- hadr_remote_service element 433
- hadr_role element 427
- hadr_standby_log_file element 435
- hadr_standby_log_lsn element 436
- hadr_standby_log_page element 436
- hadr_state element 428
- hadr_syncmode element 428
- hadr_timeout element 433
- hash join overflows monitor element 218
- hash join small overflows monitor element 218
- hash join threshold monitor element 216
- hash_join_overflows element 218
- hash_join_small_overflows element 218
- health alerts
 - enabling 507
- health center
 - health indicators 503
- Health Center
 - description 523
- Health Center Status Beacon
 - description 523
- health indicator
 - db.db_auto_storage_util 536
 - ts.ts_auto_resize_status 537
 - ts.ts_util_auto_resize 538
- health indicators
 - catalog cache hit ratio 553
 - collection state-based 503
 - data 512
 - database heap utilization 555
 - database highest severity alert state 546
 - database operational state 546
 - DBMS highest severity alert state 545
 - deadlock rate 550
 - DMS table spaces 535
 - format 533
 - identifier information
 - db.alert_state 546
 - db.apps_waiting_locks 553
 - db.catcache_hitratio 553
 - db.database_heap_utilization 555
 - db.db_backup_req 548

- health indicators (*continued*)
 - identifier information (*continued*)
 - db.db_op_status 546
 - db.deadlock_rate 550
 - db.fed_nicknames_status 556
 - db.fed_servers_status 556
 - db.hadr_delay 549
 - db.hadr_op_status 548
 - db.lock_escal_rate 552
 - db.locklist_utilization 551
 - db.log_fs_utilization 550
 - db.log_utilization 549
 - db.max_sort_shrmem_util 544
 - db.pkgcache_hitratio 554
 - db.shrworkspace_hitratio 554
 - db.sort_shrmem_util 542
 - db.spilled_sorts 543
 - db.tb_reorg_req 546
 - db.tb_runstats_req 547
 - db2.db2_alert_state 545
 - db2.db2_op_status 544
 - db2.mon_heap_utilization 555
 - db2.sort_privmem_util 541
 - instance operational state 544
 - lock escalation rate 552
 - lock list utilization 551
 - log filesystem utilization 550
 - log utilization 549
 - long term shared sort memory utilization 544
 - monitor heap utilization 555
 - overview 503
 - package cache hit ratio 554
 - percentage of applications waiting on locks 553
 - percentage of sorts that overflowed 543
 - private sort memory utilization 541
 - process cycle 505
 - shared sort memory utilization 542
 - shared workspace hit ratio 554
 - state-based 503
 - summary 533
 - table space container operational state 541
 - table space container utilization 539
 - table space operational state 540
 - table space utilization 538
 - threshold-based 503
 - ts.state 540
 - ts.utilization 538
 - tsc.state 541
 - tsc.utilization 539
- health monitor
 - API request types 519
 - CLP commands 516
 - description 503
 - graphical tools 523
 - Health Center 523
 - Health Center Status Beacon 523
 - interfaces 559
 - logical data groups 531
 - sample output 520
 - SQL table functions 514
 - starting and stopping 511
 - Web Health Center 523

- health snapshot
 - global 522
- health snapshots capturing
 - client application 516
 - CLP 515
 - SQL table functions 513
- help
 - displaying 568
 - for SQL statements 567
- host coded character set ID monitor element 447
- host database name monitor element 440
- host product/version ID monitor element 175
- host response time monitor element 468
- host_ccsid element 447
- host_db_name element 440
- host_prdid element 175
- host_response_time element 468

I

- ID of code page used by application monitor element 166
- idle_agents element 197
- inbound communication address monitor element 448
- inbound number of bytes received monitor element 449
- inbound number of bytes sent monitor element 450
- inbound_bytes_received element 449
- inbound_bytes_sent element 450
- inbound_comm_address element 448
- increase size by percent monitor element 337
- index object pages monitor element 362
- index used to reorganize the table monitor element 367
- index_object_pages element 362
- Information Center
 - updating 569
 - versions 568
 - viewing in different languages 568
- initial table space size monitor element 335
- input database alias monitor element 420
- input_db_alias element 420
- insert response time monitor element 480
- insert_sql_stmts element 475
- insert_time element 480
- inserts monitor element 475
- instance operational state health indicator 544
- int_auto_rebinds element 379
- int_commits element 380
- int_deadlock_rollbacks element 382
- int_rollbacks element 381
- int_rows_deleted element 358
- int_rows_inserted element 359
- int_rows_updated element 358
- internal automatic rebinds monitor element 379

- internal commits monitor element 380
- internal rollbacks due to deadlock monitor element 382
- internal rollbacks monitor element 381
- internal rows deleted monitor element 358
- internal rows inserted monitor element 359
- internal rows updated monitor element 358

L

- last active log file number monitor element 299
- last backup timestamp monitor element 158
- last extent moved by the rebalancer monitor element 340
- last reset timestamp monitor element 419
- Last table space resize attempt failed monitor element 337
- last_active_log element 299
- last_backup element 158
- last_over_flow time element 422
- last_reset element 419
- lob object pages monitor element 362
- lob_object_pages element 362
- loc_list_in_use monitor element 302
- local connections executing in the database manager monitor element 193
- local connections monitor element 192
- local databases
 - current connects monitor element 193
- local_cons element 192
- local_cons_in_exec element 193
- lock attributes monitor element 314
- lock count monitor element 316
- lock escalation monitor element 311
- lock escalation rate health indicator 552
- Lock Hold Count monitor element 316
- lock list utilization health indicator 551
- lock mode monitor element 306
- lock mode requested monitor element 311
- lock name monitor element 314
- lock node monitor element 309
- lock object name monitor element 308
- lock object type waited on monitor element 308
- lock release flags monitor element 315
- lock status monitor element 307
- lock timeout monitor element 321
- lock wait start timestamp monitor element 320
- lock waits monitor element 318
- lock_attributes monitor element 314
- lock_count monitor element 316
- lock_current_mode monitor element 316
- lock_escalation element 311
- lock_escals element 304
- lock_hold_count monitor element 316
- lock_mode element 306
- lock_mode_requested element 311

- lock_name monitor element 314
- lock_node element 309
- lock_object_name element 308
- lock_object_type element 308
- lock_release_flags monitor element 315
- lock_status element 307
- lock_timeout_val element 321
- lock_timeouts element 309
- lock_wait_start_time element 320
- lock_wait_time element 319
- lock_waits element 318
- locks

- current agents waiting on locks monitor element 320
- locks held monitor element 302
- total lock list memory in use monitor element 302
- total time unit of work waited on locks monitor element 320
- locks_held monitor element 302
- locks_held_top element 310
- locks_in_list element 314
- locks_waiting monitor element 320
- log being rolled forward monitor element 326
- log filesystem utilization health indicator 550
- log phase monitor element 326
- log read time monitor element 296
- log utilization health indicator 549
- log write time monitor element 295
- log_held_by_dirty_pages element 294
- log_read_time element 296
- log_reads element 291
- log_space_used element 292
- log_to_redo_for_recovery element 295
- log_write_time element 295
- log_writes element 291
- logical data groups 3, 129
 - event monitor 131
 - health monitor 531
 - snapshot monitor 97
- long object pages monitor element 363
- long term shared sort memory utilization health indicator 544
- long_object_pages element 363

M

- max_agent_overflows element 201
- max_data_received_1024 element 456
- max_data_received_128 element 453
- max_data_received_16384 element 460
- max_data_received_2048 element 457
- max_data_received_256 element 454
- max_data_received_31999 element 461
- max_data_received_4096 element 458
- max_data_received_512 element 455
- max_data_received_64000 element 462
- max_data_received_8192 element 459
- max_data_received_gt64000 element 462
- max_data_sent_1024 element 455
- max_data_sent_128 element 453
- max_data_sent_16384 element 459
- max_data_sent_2048 element 456
- max_data_sent_256 element 453
- max_data_sent_31999 element 460

- max_data_sent_4096 element 457
- max_data_sent_512 element 454
- max_data_sent_64000 element 461
- max_data_sent_8192 element 458
- max_data_sent_gt64000 element 462
- max_network_time_1_ms element
 - number of statements with network time of up to 1 ms monitor element 463
- max_network_time_100_ms element 465
- max_network_time_16_ms element 464
- max_network_time_4_ms element 463
- max_network_time_500_ms element 465
- max_network_time_gt500_ms element 466
- maximum agent overflows monitor element 201
- maximum database heap allocated monitor element 288
- maximum extent in range monitor element 348
- maximum network time for statement monitor element 466
- maximum number of agents registered monitor element 196
- maximum number of agents waiting monitor element 197
- maximum number of associated agents monitor element 200
- maximum number of concurrent connections monitor element 183, 441
- maximum number of coordinating agents monitor element 199
- maximum number of locks held monitor element 310
- maximum number of tablequeue buffers overflows monitor element 408
- maximum outbound number of bytes received monitor element 451
- maximum outbound number of bytes sent monitor element 451
- maximum page in range monitor element 347
- maximum private workspace size monitor element 285
- maximum secondary log space used monitor element 289
- maximum shared workspace size monitor element 282
- maximum size of memory pool monitor element 205
- maximum table reorganize phase monitor element 365
- maximum table space size monitor element 336
- maximum total log space used monitor element 290
- memory pool
 - monitor elements 202
- memory pool identifier monitor element 203
- memory pool secondary identifier monitor element 204
- memory pool watermark monitor element 206
- memory requirements
 - database system monitor 9
- message monitor element 426
- message_time monitor element 426
- minimum channels free monitor element 220
- minimum fcm buffers free monitor element 219
- minimum network time for statement monitor element 467
- minimum outbound number of bytes received monitor element 452
- minimum outbound number of bytes sent monitor element 452
- minimum recovery time until rollforward monitor element 343
- mon_heap_sz configuration parameter 9
- monitor data organization 3
- monitor element
 - active_hash_joins 215
 - data_partition_id 188
 - xquery_stmts 383
- monitor elements
 - agents and connections 190
 - application identification and status 162
 - buffer pool activity 229
 - catalog cache 273
 - container status 343
 - CPU usage 412
 - database and application activity 301
 - database configuration 229
 - database heap 288
 - database identification and status 153
 - database manger configuration 189
 - database system 147
 - DB2 agent information 188
 - DB2 connect 437
 - dynamic SQL 409
 - event monitoring 421
 - fast communications manager 219
 - federated database systems 474
 - hash joins 215
 - high availability disaster recovery 427
 - intra-query parallelism 411
 - lock wait information 317
 - locks and deadlocks 301
 - logging 288
 - memory pool 202
 - non-buffered I/O activity 268
 - package cache 277
 - server identification and status 148
 - snapshot monitor logical data groups 100
 - snapshot monitoring 419
 - sort 207
 - SQL cursors 369
 - SQL statement activity 372
 - SQL statement details 384
 - SQL workspaces 282
 - subsection details 403
 - table activity 350
 - table reorganization 363
 - table space activity 327
 - table space quiescer activity 341
 - table space range status 346
 - transaction processor monitoring 472
- monitor elements (*continued*)
 - utilities 222
- monitor heap utilization health indicators 555
- monitor switches
 - description 13
 - setting from a client application 17
 - setting from the CLP 15
- monitoring
 - capturing a snapshot from client applications 46
 - capturing a snapshot from the command line 43
 - capturing a snapshot using SQL 22, 57
 - with file access 27
 - with SNAP_WRITE_FILE 25
 - data partitions 29
 - database activity 488, 492
 - database events 59
 - health monitor 503, 511
 - open access to monitor data
 - capturing snapshot information to a file 25
 - retrieving snapshot information from a file 27
 - SYSMON authority 22
 - system monitor 3
- most recent connection elapsed time monitor element 470
- most recent response time for connect monitor element 470
- most recent statement elapsed time monitor element 392
- most recent unit of work elapsed time monitor element 186

N

- network_time_bottom element 467
- network_time_top element 466
- new buffer pool size monitor element 268
- nickname status health indicator 556
- node number monitor element 181
- node with least available log space monitor element 168
- node_number element 181
- nonblocked event monitors 74
- notices 573
- num_agents element 411
- num_assoc_agents element 201
- num_block_IOs element 258
- num_compilation element 410
- num_db_storage_paths element 159
- num_executions element 409
- num_gw_conn_switches element 202
- num_indoubt_trans element 317
- num_log_buffer_full element 298
- num_log_data_found_in_buffer element 298
- num_log_part_page_io element 297
- num_log_read_io element 297
- num_log_write_io element 296
- num_nodes_in_db2_instance element 420

num_pages_from_block_IOs element 259
 num_pages_from_vectored_IOs element 257
 num_transmissions element 469
 num_transmissions_group element 469
 num_vectored_IOs element 257
 number of agents created monitor element 412
 number of agents working on a statement monitor element 411
 number of associated agents monitor element 201
 number of automatic storage paths monitor element 159
 number of block IO requests monitor element 258
 number of connections waiting for the client to send request monitor element 443
 number of connections waiting for the host to reply monitor element 442
 number of containers in range monitor element 349
 number of containers in tablespace monitor element 343
 number of event monitor activations monitor element 425
 number of event monitor flushes monitor element 424
 number of event monitor overflows monitor element 421
 number of extents the rebalancer has processed monitor element 339
 number of full log buffers monitor element 298
 number of idle agents monitor element 197
 number of indoubt transactions monitor element 317
 number of lock escalations monitor element 304
 number of lock timeouts monitor element 309
 number of locks reported monitor element 314
 number of log data found in buffer monitor element 298
 number of log pages read monitor element 291
 number of log pages written monitor element 291
 number of log reads monitor element 297
 number of log writes monitor element 296
 number of nodes in partition monitor element 420
 number of open cursors monitor element 445
 number of pages left to remove monitor element 268
 number of partial log page writes monitor element 297
 number of physical page maps monitor element 259
 number of quiescers monitor element 340
 number of ranges in the tablespace map monitor element 346
 number of rows read from tablequeues monitor element 407
 number of rows written to tablequeues monitor element 408
 number of SQL chains attempted monitor element 444
 number of SQL statements attempted monitor element 444
 number of statements with network time between 100 and 500 ms monitor element 465
 number of statements with network time between 16 and 100 ms monitor element 465
 number of statements with network time between 2 and 4 ms monitor element 463
 number of statements with network time between 8 and 16 ms monitor element 464
 number of statements with network time greater than 500 ms monitor element 466
 number of statements with network time of up to 1 ms monitor element 463
 number of statements with outbound bytes received between 1 and 128 bytes monitor element 453
 number of statements with outbound bytes received between 1025 and 2048 bytes monitor element 457
 number of statements with outbound bytes received between 129 and 256 bytes monitor element 454
 number of statements with outbound bytes received between 16385 and 31999 bytes monitor element 461
 number of statements with outbound bytes received between 2049 and 4096 bytes monitor element 458
 number of statements with outbound bytes received between 257 and 512 bytes monitor element 455
 number of statements with outbound bytes received between 32000 and 64000 bytes monitor element 462
 number of statements with outbound bytes received between 4097 and 8192 bytes monitor element 459
 number of statements with outbound bytes received between 513 and 1024 bytes monitor element 456
 number of statements with outbound bytes received between 8193 and 16384 bytes monitor element 460
 number of statements with outbound bytes received greater than 64000 bytes monitor element 462
 number of statements with outbound bytes sent between 1 and 128 bytes monitor element 453
 number of statements with outbound bytes sent between 1025 and 2048 bytes monitor element 456
 number of statements with outbound bytes sent between 129 and 256 bytes monitor element 453
 number of statements with outbound bytes sent between 16385 and 31999 bytes monitor element 460
 number of statements with outbound bytes sent between 2049 and 4096 bytes monitor element 457
 number of statements with outbound bytes sent between 257 and 512 bytes monitor element 454
 number of statements with outbound bytes sent between 32000 and 64000 bytes monitor element 461
 number of statements with outbound bytes sent between 4097 and 8192 bytes monitor element 458
 number of statements with outbound bytes sent between 513 and 1024 bytes monitor element 455
 number of statements with outbound bytes sent between 8193 and 16384 bytes monitor element 459
 number of statements with outbound bytes sent greater than 64000 bytes monitor element 462
 number of successful fetches monitor element 394
 number of table spaces mapped to buffer pool monitor element 268
 number of transmissions group monitor element 469
 number of transmissions monitor element 469
 number of vectored io requests monitor element 257

O

open local cursors monitor element 371
 open local cursors with blocking monitor element 372
 open remote cursors monitor element 369
 open remote cursors with blocking monitor element 369
 open_cursors element 445
 open_loc_curs element 371
 open_loc_curs_blk element 372
 open_rem_curs element 369
 open_rem_curs_blk element 369
 operation elements 386
 ordering DB2 books 566
 original lock mode before conversion monitor element 316
 outbound application ID monitor element 175
 outbound communication address monitor element 448
 outbound communication protocol monitor element 447
 outbound number of bytes received monitor element 450

- outbound number of bytes sent monitor element 449
- outbound sequence number monitor element 176
- outbound_appl_id element 175
- outbound_bytes_received element 450
- outbound_bytes_received_bottom element 452
- outbound_bytes_received_top element 451
- outbound_bytes_sent element 449
- outbound_bytes_sent_bottom element 452
- outbound_bytes_sent_top element 451
- outbound_comm_address element 448
- outbound_comm_protocol element 447
- outbound_sequence_no element 176
- overflow_accesses element 357

P

- package cache high water mark monitor element 280
- package cache hit ratio health indicator 554
- package cache inserts monitor element 279
- package cache lookups monitor element 277
- package cache overflows monitor element 279
- package consistency token monitor element 388
- package name monitor element 387
- package version monitor element 388
- package_name element 387
- package_version_id element 388
- page reorganizations monitor element 360
- page_reorgs element 360
- partial record monitor element 423
- partial_record element 423
- participant holding a lock on the object required by application monitor element 313
- participant within deadlock monitor element 313
- participant_no element 313
- participant_no_holding_lk element 313
- partition number monitor element 426
- partition number where deadlock occurred monitor element 312
- partition_number monitor element 426
- partitioned database environments
 - global snapshots 53
- partitioned databases
 - event monitoring 77
- partitioned tables
 - reorganizing 29
- pass-through monitor element 477
- pass-through time monitor element 482
- passthru element 477
- passthru_time element 482
- pending free pages in table space monitor element 334
- percentage of applications waiting on locks health indicator 553

- percentage of sorts that overflowed health indicator 543
- physical_page_maps element 259
- pipe event monitors
 - creating 75
 - formatting output from command line 79
 - named pipe management 76
- pipeds sorts accepted monitor element 209
- pipeds sorts requested monitor element 208
- pipeds_sorts_accepted element 209
- pipeds_sorts_requested element 208
- pkg_cache_inserts element 279
- pkg_cache_lookups element 277
- pkg_cache_num_overflow element 279
- pkg_cache_size_top element 280
- pool_async_data_read_reqs element 250
- pool_async_data_reads element 245
- pool_async_data_writes element 246
- pool_async_index_read_reqs element 251
- pool_async_index_reads element 248
- pool_async_index_writes element 247
- pool_async_read_time element 249
- pool_async_write_time element 249
- pool_async_xda_read_reqs 259
- pool_async_xda_reads 260
- pool_async_xda_writes 261
- pool_cur_size element 204
- pool_data_l_reads element 233
- pool_data_p_reads element 235
- pool_data_writes element 237
- pool_drty_pg_steal_clns element 252
- pool_drty_pg_thrsh_clns element 254
- pool_id element 203
- pool_index_l_reads element 238
- pool_index_p_reads element 240
- pool_index_writes element 241
- pool_lsn_gap_clns element 251
- pool_max_size element 205
- pool_no_victim_buffer element 254
- pool_read_time element 243
- pool_secondary_id monitor element 204
- pool_temp_data_l_reads element 234
- pool_temp_data_p_reads element 236
- pool_temp_index_l_reads element 239
- pool_temp_index_p_reads element 241
- pool_temp_xda_l_reads 265
- pool_temp_xda_p_reads 266
- pool_watermark element 206
- pool_write_time element 243
- pool_xda_l_reads 262
- pool_xda_p_reads 263
- pool_xda_writes 264
- post shared threshold sorts monitor element 214
- post threshold hash joins monitor element 216
- post threshold sorts monitor element 208
- post_shrthreshold_hash_joins monitor element 216
- post_shrthreshold_sorts monitor element 214
- post_threshold_hash_joins element 216

- post_threshold_sorts element 208
- prefetch_wait_time element 256
- prep_time_best element 410
- prep_time_worst element 410
- prev_uow_stop_time element 184
- previous unit of work completion timestamp monitor element 184
- printed books
 - ordering 566
- priv_workspace_num_overflows element 285
- priv_workspace_section_inserts element 287
- priv_workspace_section_lookups element 286
- priv_workspace_size_top element 285
- private sort memory utilization health indicator 541
- private workspace overflows monitor element 285
- private workspace section inserts monitor element 287
- private workspace section lookups monitor element 286
- problem determination
 - online information 571
 - tutorials 571
- process or thread id monitor element 189
- product name monitor element 152
- product_name monitor element 152
- progress description monitor element 226
- progress sequence number monitor element 225
- progress start time monitor element 226
- progress work metric monitor element 226
- progress_completed_units element 227
- progress_description element 226
- progress_list_attr monitor element 225
- progress_list_cur_seq_num element 225
- progress_seq_num element 225
- progress_start_time element 226
- progress_total_units element 227
- progress_work_metric element 226

Q

- query cost estimate monitor element 396
- query number of rows estimate monitor element 395
- query response time monitor element 479
- query_card_estimate element 395
- query_cost_estimate monitor element 396
- quiescer monitor elements
 - quiescer agent identification monitor element 341
 - quiescer object identification monitor element 342
 - quiescer state monitor element 342
 - quiescer tablespace identification monitor element 341
 - quiescer user authorization identification monitor element 341

quiescer_agent_id element 341
quiescer_auth_id element 341
quiescer_obj_id element 342
quiescer_state element 342
quiescer_ts_id element 341

R

range adjustment monitor element 348
range container monitor element 349
range number monitor element 347
range offset monitor element 349
range_adjustment element 348
range_container_id element 349
range_end_stripe element 348
range_max_extent element 348
range_max_page_number element 347
range_num_containers element 349
range_number element 347
range_offset element 349
range_start_stripe element 348
range_stripe_set_number element 347
rebalancer mode monitor element 338
rebalancer restart time monitor element 339
rebalancer start time monitor element 338
rej_curs_blk element 370
rejected block cursor requests monitor element 370
rem_cons_in element 191
rem_cons_in_exec element 192
remote connections executing in the database manager monitor element 192
remote connections to database manager monitor element 191
remote lock time monitor element 483
remote locks monitor element 478
remote_lock_time element 483
remote_locks element 478
reorg_completion element 366
reorg_current_counter element 366
reorg_end element 367
reorg_index_id monitor element 367
reorg_long_tbspc_id monitor element 368
reorg_max_counter element 366
reorg_max_phase element 365
reorg_phase monitor element 365
reorg_phase_start element 365
reorg_rows_compressed monitor element 368
reorg_rows_rejected_for_compression monitor element 368
reorg_start element 367
reorg_status element 364
reorg_tbspc_id monitor element 367
reorg_type element 363
reorganization required health indicator 546
reorganize phase monitor element 365
request identifier for sql statement monitor element 425
rf_log_num element 326
rf_status element 326
rf_timestamp element 325

rf_type element 326
rollback statements attempted monitor element 376
rollback_sql_stmts element 376
rolled back agent monitor element 323
rolled back application monitor element 323
rolled back application participant monitor element 313
rolled back sequence number monitor element 324
rolled_back_agent_id element 323
rolled_back_appl_id element 323
rolled_back_participant_no element 313
rolled_back_sequence_no element 324
rollforward timestamp monitor element 325
rollforward type monitor element 326
rows compressed monitor element 368
rows deleted monitor element 353
rows inserted monitor element 353
rows read monitor element 356
rows rejected for compression monitor element 368
rows returned by stored procedures monitor element 479
rows selected monitor element 355
rows updated monitor element 354
rows written monitor element 355
rows_deleted element 353
rows_inserted element 353
rows_read element 356
rows_selected element 355
rows_updated element 354
rows_written element 355

S

sec_log_used_top element 289
sec_logs_allocated element 290
secondary connections monitor element 200
secondary logs allocated currently monitor element 290
section inserts monitor element 281
section lookups monitor element 280
section number monitor element 389
section_number element 389
select SQL statements executed monitor element 377
select_sql_stmts element 377
select_time element 479
self-describing data stream
 database system monitor 8
 event monitors 91
 snapshot monitor 54
 system monitor switches 19
sequence number holding lock monitor element 323
sequence number monitor element 172
sequence_no element 172
sequence_no_holding_lk element 323
server instance name monitor element 149
server operating system monitor element 152
server product/version ID monitor element 150
server version monitor element 151
server_db2_type element 150
server_instance_name element 149
server_nname element 149
server_platform element 152
server_prdid element 150
server_version element 151
service level monitor element 151
service_level monitor element 151
session authorization ID monitor element 173
session_auth_id element 173
shared sort memory utilization health indicator 542
shared workspace hit ratio health indicator 554
shared workspace overflows monitor element 283
shared workspace section inserts monitor element 284
shared workspace section lookups monitor element 283
shr_workspace_num_overflows element 283
shr_workspace_section_inserts element 284
shr_workspace_section_lookups element 283
shr_workspace_size_top element 282
smallest_log_avail_node element 168
snapshot monitoring
 capturing
 using SQL with file access 27
 capturing to file 25
 description 21
 interpreting output for data partitions 29
 making snapshot data available for all users 25
 on data partitions 29
 on partitioned database systems 53
 output
 self-describing data stream 54
 SQL table functions 37
 subsections 52
 using a client application 46
 using CLP 43
 using SQL 57
 using SQL with direct access 22
 with SNAP_WRITE_FILE 25
snapshot time monitor element 420
snapshots
 capturing
 using SQL with file access 27
 capturing to file 25
 capturing with
 SNAP_WRITE_FILE 25
 making snapshot data available for all users 25
 SQL table functions 37
 using SQL with direct access 22
sort overflows monitor element 211
Sort Private Heap High Water Mark monitor element 213

- sort share heap currently allocated monitor element 213
- Sort Share Heap High Water Mark monitor element 213
- sort_heap_allocated element 207
- sort_heap_top monitor element 213
- sort_overflows element 211
- sort_shrheap_allocated monitor element 213
- sort_shrheap_top monitor element 213
- sorting
 - monitor elements 207
- sp_rows_selected element 479
- SQL communications area (SQLCA) monitor element 395
- SQL cursors
 - monitor elements 369
- SQL dynamic statement text monitor element 393
- SQL requests since last commit monitor element 382
- SQL statements
 - displaying help 567
- SQL statements monitor element 378
- SQL table functions
 - capturing health snapshots 513
 - health monitor 514
- SQL workspaces
 - monitor elements 282
- sql_chains element 444
- sql_req_id element 425
- sql_reqs_since_commit element 382
- sql_stmts element 444
- sqlca element 395
- ss_exec_time element 405
- ss_node_number element 404
- ss_number element 404
- ss_status element 404
- ss_sys_cpu_time element 418
- ss_usr_cpu_time element 417
- start database manager timestamp monitor element 148
- start stripe monitor element 348
- start_time element 392
- state change object identification monitor element 342
- state change tablespace identification monitor element 343
- state-based health indicators 503
- statement best preparation time monitor element 410
- statement compilations monitor element 410
- statement execution elapsed time monitor element 467
- statement executions monitor element 409
- statement first use time monitor element 397
- statement history identifier monitor element 397
- statement history list size monitor element 402
- statement invocation identifier monitor element 399
- statement isolation monitor element 398
- statement last use time monitor element 398
- statement lock timeout monitor element 398
- statement nesting level monitor element 399
- statement node monitor element 383
- statement operation monitor element 386
- statement operation start timestamp monitor element 391
- statement operation stop timestamp monitor element 391
- statement package cache identifier monitor element 400
- statement query identifier monitor element 400
- statement sorts monitor element 394
- statement source identifier monitor element 400
- statement type monitor element 385
- statement worst preparation time monitor element 410
- static SQL statements attempted monitor element 373
- static_sql_stmts element 373
- statistics collection required health indicator 547
- status of database monitor element 156
- status of DB2 instance monitor element 152
- status_change_time element 167
- stmt_elapsed_time element 392
- stmt_first_use_time element 397
- stmt_history_id element 397
- stmt_history_list_size element 402
- stmt_invocation_id element 399
- stmt_isolation element 398
- stmt_last_use_time 398
- stmt_lock_timeout element 398
- stmt_nest_level element 399
- stmt_node_number element 383
- stmt_operation element 386
- stmt_pkgcache_id element 400
- stmt_query_id element 400
- stmt_sorts element 394
- stmt_source_id element 400
- stmt_start element 391
- stmt_stop element 391
- stmt_sys_cpu_time element 415
- stmt_text element 393
- stmt_type element 385
- stmt_usr_cpu_time element 414
- stmt_value_data element 402
- stmt_value_index element 402
- stmt_value_isnull element 401
- stmt_value_isreoptvalue element 403
- stmt_value_type element 401
- stolen agents monitor element 199
- stop_time element 391
- stored procedure time monitor element 482
- stored procedures monitor element 478
- stored_proc_time element 482
- stored_procs element 478
- stripe set monitor element 345
- stripe set number monitor element 347
- subsection execution elapsed time monitor element 405
- subsection node number monitor element 404
- subsection number monitor element 404
- subsection snapshots 52
- subsection status monitor element 404
- summary, health indicators 533
- SYSMON authority 22
- system CPU time monitor element 416
- system CPU time used by agent monitor element 413
- system CPU time used by statement monitor element 415
- system CPU time used by subsection monitor element 418
- system monitor 3
- system monitor switches
 - description 13
 - self-describing data stream 19
 - setting from a client application 17
 - setting from the CLP 15
 - types 13
- system_cpu_time element 416

T

- table event monitors
 - creating 64
 - table management 67
- table file ID monitor element 360
- table name monitor element 351
- table reorganization
 - monitor elements 363
- table reorganize attribute flag monitor element 363
- table reorganize completion flag monitor element 366
- table reorganize end time monitor element 367
- table reorganize phase start time monitor element 365
- table reorganize start time monitor element 367
- table reorganize status monitor element 364
- table schema name monitor element 352
- Table space auto-resize enabled monitor element 335
- table space container operational state health indicator 541
- table space container utilization health indicator 539
- table space extent size monitor element 331
- table space high water mark monitor element 334
- table space increase size in bytes monitor element 336
- table space name monitor element 328
- table space operational state health indicator 540
- table space page size monitor element 331
- table space prefetch size monitor element 331

table space quiescer activity
 monitor elements 341
 table space state monitor element 330
 Table space using automatic storage
 monitor element 334
 table space utilization health
 indicator 538
 table space where long objects are
 reorganized monitor element 368
 table space where table or data partition
 is reorganized monitor element 367
 table type monitor element 350
 table_file_id element 360
 table_name element 351
 table_schema element 352
 table_type element 350
 tablespace being rolled forward monitor
 element 325
 tablespace contents type monitor
 element 330
 tablespace identification monitor
 element 328
 tablespace type monitor element 329
 tablespace_auto_resize_enabled
 element 335
 tablespace_content_type element 330
 tablespace_cur_pool_id element 332
 tablespace_current_size element 335
 tablespace_extent_size element 331
 tablespace_free_pages element 334
 tablespace_id element 328
 tablespace_increase_size element 336
 tablespace_increase_size_percent
 element 337
 tablespace_initial_size 335
 tablespace_last_resize_failed 337
 tablespace_last_resize_time element 337
 tablespace_max_size element 336
 tablespace_min_recovery_time
 element 343
 tablespace_name element 328
 tablespace_next_pool_id element 332
 tablespace_num_containers element 343
 tablespace_num_quiescers element 340
 tablespace_num_ranges element 346
 tablespace_page_size element 331
 tablespace_page_top monitor
 element 334
 tablespace_pending_free_pages
 element 334
 tablespace_prefetch_size element 331
 tablespace_rebalancer_extents_processed
 element 339
 tablespace_rebalancer_extents_remaining
 element 339
 tablespace_rebalancer_last_extent_moved
 element 340
 tablespace_rebalancer_mode
 element 338
 tablespace_rebalancer_priority
 element 340
 tablespace_rebalancer_restart_time
 element 339
 tablespace_rebalancer_start_time
 element 338
 tablespace_state element 330
 tablespace_state_change_object_id
 element 342
 tablespace_state_change_ts_id
 element 343
 tablespace_total_pages element 332
 tablespace_type element 329
 tablespace_usable_pages element 333
 tablespace_used_pages element 333
 tablespace_using_auto_storage
 element 334
 terms and conditions
 use of publications 572
 threshold-based health indicators 503
 time of database connection monitor
 element 156
 time of first event overflow monitor
 element 422
 time of last event overflow monitor
 element 422
 Time of last successful table space resize
 monitor element monitor element 337
 time waited for prefetch monitor
 element 256
 time waited on locks monitor
 element 319
 time zone displacement monitor
 element 153
 time_stamp element 420
 time_zone_disp element 153
 timestamp control table message monitor
 element 426
 tot_log_used_top element 290
 tot_s_cpu_time element 418
 tot_u_cpu_time element 419
 total buffer pool physical read time
 monitor element 243
 total buffer pool physical write time
 monitor element 243
 total fcm buffers received monitor
 element 222
 total fcm buffers sent monitor
 element 221
 total hash joins monitor element 215
 total hash loops monitor element 217
 total log available monitor element 293
 total log space used monitor
 element 292
 total number of attempted connections
 for DB2 Connect monitor element 441
 total number of extents to be processed
 by the rebalancer monitor element 339
 total number of pages in object monitor
 element 366
 total number of pages read by block I/O
 monitor element 259
 total number of pages read by vectored
 I/O monitor element 257
 total number of tablequeue buffers
 overflowed monitor element 406
 total pages in container monitor
 element 345
 total pages in table space monitor
 element 332
 total progress work units monitor
 element 227
 Total size of a file system monitor
 element 160
 total sort heap allocated monitor
 element 207
 total sort time monitor element 210
 total sorts monitor element 210
 total system CPU for a statement monitor
 element 418
 total user CPU for a statement monitor
 element 419
 total_buffers_rcvd element 222
 total_buffers_sent element 221
 total_cons element 194
 total_exec_time element 411
 total_hash_joins element 215
 total_hash_loops element 217
 total_log_available element 293
 total_log_used element 292
 total_sec_cons element 200
 total_sort_time element 210
 total_sorts element 210
 TP monitor client accounting string
 monitor element 474
 TP monitor client application name
 monitor element 473
 TP monitor client user ID monitor
 element 472
 TP monitor client workstation name
 monitor element 473
 tpmon_acc_str element 474
 tpmon_client_app element 473
 tpmon_client_userid element 472
 tpmon_client_wkstn element 473
 tq_cur_send_spills element 407
 tq_id_waiting_on element 409
 tq_max_send_spills element 408
 tq_node_waited_for element 406
 tq_rows_read element 407
 tq_rows_written element 408
 tq_tot_send_spills element 406
 tq_wait_for_any element 405
 transaction ID monitor element 467
 troubleshooting
 online information 571
 tutorials 571
 ts_name element 325
 ts.state health indicators 540
 ts.ts_auto_resize_status 537
 ts.ts_util_auto_resize 538
 ts.utilization health indicator 538
 tsc.state health indicator 541
 tsc.utilization health indicator 539
 tutorials
 troubleshooting and problem
 determination 571
 Visual Explain 571

U

uid_sql_stmts element 378
 Unique file system id number monitor
 element 161
 unit of work completion status monitor
 element 186
 unit of work log space used monitor
 element 292
 unit of work start timestamp monitor
 element 185
 unit of work status monitor element 187

- unit of work stop timestamp monitor element 185
- unread prefetch pages monitor element 256
- unread_prefetch_pages element 256
- uow_comp_status element 186
- uow_elapsed_time element 186
- uow_lock_wait_time monitor element 320
- uow_log_space_used element 292
- uow_start_time element 185
- uow_status element 187
- uow_stop_time element 185
- update response time monitor element 480
- update_sql_stmts element 476
- update_time element 480
- update/insert/delete SQL statements executed monitor element 378
- updates
 - DB2 Information Center 569
 - Information Center 569
- updates monitor element 476
- usable pages in container monitor element 345
- usable pages in table space monitor element 333
- used pages in table space monitor element 333
- user authorization level monitor element 180
- user CPU time monitor element 416
- user CPU time used by agent monitor element 413
- user CPU time used by statement monitor element 414
- user CPU time used by subsection monitor element 417
- user login ID monitor element 176
- user_cpu_time element 416
- utility description monitor element 224
- utility id monitor element 223
- utility invoker type monitor element 229
- utility priority monitor element 224
- utility start time monitor element 224
- utility state monitor element 228
- utility type monitor element 223
- utility_dbname element 223
- utility_description element 224
- utility_id element 223
- utility_invoker_type element 229
- utility_priority element 224
- utility_start_time element 224
- utility_state element 228
- utility_type element 223

V

- value data monitor element 402
- value has null value monitor element 401
- value index monitor element 402
- value type monitor element 401
- Variable used for statement reoptimization monitor element 403
- version monitor element 423

- version of monitor data monitor element 423
- Visual Explain tutorial 571

W

- Waited for Node on a Tablequeue monitor element 406
- Waited on Node on a Tablequeue monitor element 409
- Waiting for Any Node to Send on a tablequeue monitor element 405
- Web Health Center 523
- write-to-table event monitors, buffering 74

X

- x_lock_escals element 305
- xda object pages monitor element 267
- xda_object_pages 267
- xid element 467
- xquery_stmts monitor element 383

Contacting IBM

To contact IBM in your country or region, check the IBM Directory of Worldwide Contacts at <http://www.ibm.com/planetwide>

To learn more about DB2 products, go to <http://www.ibm.com/software/data/db2/>.



Printed in USA

SC10-4251-00



Spine information:

IBM DB2 DB2 Version 9

System Monitor Guide and Reference

