IBM® DB2 Universal Database™

System Monitor Guide and Reference

Version 8

SC09-4847-00

IBM® DB2 Universal Database™

System Monitor Guide and Reference

Version 8

SC09-4847-00

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

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.

Contents

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:

- "Event monitors" on page 45
- "Counter status and visibility" on page 5
- "Database system monitor data organization" on page 4
- "Database system monitor memory requirements" on page 7
- "Snapshot monitor" on page 21

Database system monitor data organization

The database system monitor stores information it collects in entities called *monitor elements* (these are also 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 are resettable.
- 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 are not resettable.
- Water mark: indicates the highest (maximum) or lowest (minimum) value an element has reached since monitoring was started. Water mark elements are not resettable.
- Information: provides reference-type details of your monitoring activities. This can include items such as partition names, aliases, and path details. Information elements are not resettable.
- Timestamp: indicates the date and time that an activity took place by providing the number of seconds and microseconds that have elapsed since January 1, 1970. For the snapshot monitor and event monitors, the collection of timestamp elements is controlled by the TIMESTAMP monitor switch. While this switch is on by default, you should turn it off for performance reasons if CPU utilization on the database instance approaches 100%. Timestamp elements are not resettable.
- Time: returns the number of seconds and microseconds spent on an activity. For the snapshot monitor and event monitors, the collection of most time elements is controlled by the TIMESTAMP monitor switch. While this switch is on by default, you should turn it off for performance reasons if CPU utilization on the database instance approaches 100%. Some time elements are resettable.

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:

- "Database system monitor" on page 3
- "System monitor switches" on page 11
- "Event monitors" on page 45
- "Counter status and visibility" on page 5
- "Snapshot monitor" on page 21

Related reference:

- "RESET MONITOR Command" in the *Command Reference*

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 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.

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.

Related concepts:

- "Database system monitor" on page 3
- "System monitor switches" on page 11
- "Event monitors" on page 45
- "Database system monitor data organization" on page 4
- "Snapshot monitor" on page 21

Related reference:

- "RESET MONITOR Command" in the *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 (*db*) contains individual monitor elements (for example, *total_log_used*) along with other logical data groupings, such as rollforward information (*rollforward*). This does not include the size taken up by the 'size', 'type', and 'element' information.

type The type of element stored in the data (for example, variable length string or signed 32 bit numeric value). An element type of *header* refers to a logical data grouping for an element.

element id The identifier for the monitor element that was captured by

the monitor. In the case of a logical data grouping, this is the identifier for the group (for example, *collected*, *dbase*, or *event_db*).

data The value collected by a monitor for a monitor element. In the case of a logical data grouping, the data is composed of the monitor elements belonging to it.

All timestamps in monitor elements are returned in two unsigned 4 byte monitor elements (seconds and microseconds). These represent the number of seconds since January 1, 1970 in GMT time.

The size element of strings in monitor elements represents the actual size of data for the string element. This size does not include a null terminator, as the strings are not null terminated.

Related concepts:

- "Event type mappings to logical data groups" on page 132
- "Snapshot monitor self-describing data stream" on page 41
- "Event monitor self-describing data stream" on page 81
- "Monitor switches self-describing data stream" on page 19

Related reference:

- "Snapshot monitor interface mappings to logical data groups" on page 89

Database system monitor memory requirements

The memory required for maintaining database system monitor data is allocated from the monitor heap. Its 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, and 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:

```
(Storage used by applications           +
 Storage used by event monitors         +
 Storage used by monitoring applications +
 Storage used by Gateway applications)      / 4096
```

Storage used by each application:

- If the STATEMENT SWITCH is off, zero

- If the STATEMENT switch is on:
  - Add 400 bytes for each statement being run at the same time (That is, the number of open cursors that an application might have). This is NOT the cumulative total of statements an application has run.
  - If a partitioned database, add the following for each statement:
    - 200 bytes * (average # of subsections)
- If the application has issued sqleseti() info, add the sizes of the userid, applname, workstation name and accounting string.

Storage used by each event monitor:

- 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:
  - add 600 bytes
  - for each connected application:
    - add 600 bytes
  - remember to add the ″Storage used by applications″ above

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 11

Related reference:

- "Database System Monitor Heap Size configuration parameter - mon_heap_sz" in the *Administration Guide: Performance*

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 ON in one application's configuration, then the database manager always collects that monitor data. If the same switch is then set OFF in the application's configuration, then the database manager will still collect data as long as there is at least one application with this switch turned ON.

The collection of time and timestamp elements is controlled by the TIMESTAMP switch. Turning this switch OFF (it is ON by default) instructs the database manager to skip any timestamp operating system calls when determining time or timestamp-related monitor elements. Turning this switch OFF becomes important as CPU utilization approaches 100%. When this occurs, the performance degradation caused by issuing timestamps increases dramatically. For monitor elements that can be controlled by the TIMESTAMP switch and another switch, if either of the switches is turned OFF, data is not collected. Therefore, if the TIMESTAMP switch is turned OFF, the overall cost of data under the control of other monitor switches is greatly reduced.

Event monitors are not affected by monitor switches in the same way as snapshot monitoring applications. When an event monitor is defined, it automatically turns ON the instance level monitor switches required by the specified event types. For example, a deadlock event monitor will automatically turn ON the LOCK monitor switch. The required monitor switches are turned ON when the even 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 1. Snapshot Monitor Switches

Monitor Switch	DBM Parameter	Information Provided
BUFFERPOOL	DFT_MON_BUFPOOL	Number of reads and writes, time taken
LOCK	DFT_MON_LOCK	Lock wait times, deadlocks
SORT	DFT_MON_SORT	Number of heaps used, sort performance

Table 1. Snapshot Monitor Switches (continued)

Monitor Switch	DBM Parameter	Information Provided
STATEMENT	DFT_MON_STMT	Start/stop time, statement identification
TABLE	DFT_MON_TABLE	Measure of activity (rows read/written)
UOW	DFT_MON_UOW	Start/end times, completion status
TIMESTAMP	DFT_MON_TIMESTAMP	Timestamps

Related concepts:

- "Event monitors" on page 45
- "Snapshot monitor" on page 21
- "Monitor switches self-describing data stream" on page 19

Related tasks:

- "Setting monitor switches from a client application" on page 16
- "Setting monitor switches from the CLP" on page 13

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, or SYSMAINT 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:

- 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
  ```
- To check the status of the local monitor switches use the GET MONITOR SWITCHES command.

  ```
  db2 get monitor switches
  ```

  The following is an example of the output you would expect to see after issuing the above UPDATE MONITOR SWITCH command:

  ```
           Monitor Recording Switches

  Switch list for db partition number 1
  Buffer Pool Activity Information (BUFFERPOOL) = OFF
  Lock Information                         (LOCK) = OFF
  Sorting Information                      (SORT) = OFF
  SQL Statement Information           (STATEMENT) = OFF
  Table Activity Information              (TABLE) = OFF
  Unit of Work Information                  (UOW) = OFF
  Get timestamp information           (TIMESTAMP) = OFF
  ```

- It is also possible to manipulate the monitor switches at the database manager level. This involves changing the dft_monswitches parameters in the database manager configuration file, using the UPDATE DBM CFG command.

  ```
  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.
- 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
  ```
- 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
  ```
- 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
```

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:

- "System monitor switches" on page 11
- "Event monitors" on page 45
- "Snapshot monitor" on page 21

Related tasks:

- "Setting monitor switches from a client application" on page 16

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, or SYSMAINT authority to use the db2MonitorSwitches API.

Procedure:

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

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

2. Set switch lists buffer unit size to 10 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 (&switchesList, '\0', sizeof(switchesList));
   struct sqlm_recording_group switchesList[SQLM_NUM_GROUPS];
   memset (&sqlm_recording_group, '\0',
           sizeof(struct sqlm_recording_group));
   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', sizeof(switchesBuffer));
   ```

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:

- "System monitor switches" on page 11
- "Event monitors" on page 45
- "Snapshot monitor" on page 21
- "Monitor switches self-describing data stream" on page 19

Related tasks:

- "Setting monitor switches from the CLP" on page 13

Related reference:

- "db2MonitorSwitches - Get/Update Monitor Switches" in the *Administrative API Reference*

Related samples:

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

- "dbsnap.out -- HOW TO GET A SNAPSHOT AT DATABASE LEVEL (C++)"
- "insnap.C -- Capture a snapshot at the instance level (C++)"
- "insnap.out -- HOW TO GET A SNAPSHOT AT INSTANCE LEVEL (C++)"
- "utilsnap.C -- Utilities for the snapshot monitor samples (C++)"
- "dbsnap.cbl -- Get a database monitor snapshot (IBM COBOL)"

Monitor switches self-describing data stream

After you update and/or view the current monitor switch settings with the db2MonitorSwitches API, it 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. 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.
2. For global switch requests the partition order of the returned information can be different in each switch request. In this case, a partition id is included in the data stream.

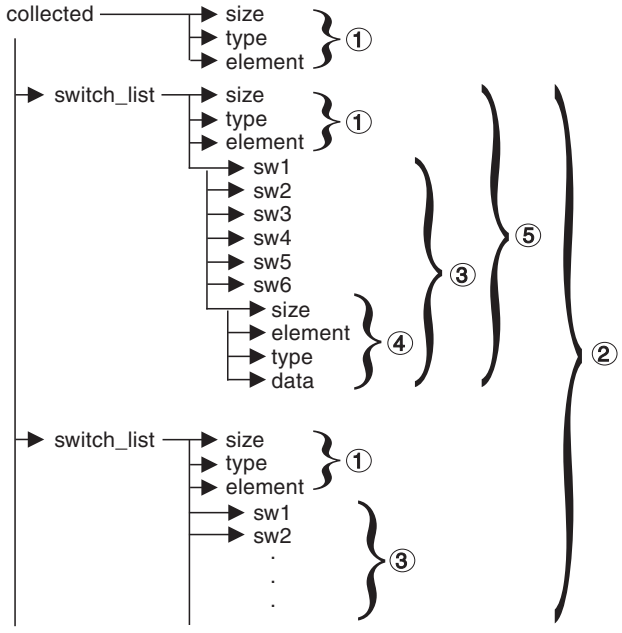


Figure 1. Switch List Monitor Data Stream

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

Related concepts:

- "System monitor switches" on page 11
- "System monitor output: the self-describing data stream" on page 6

Related tasks:

- "Setting monitor switches from a client application" on page 16

Related reference:

- "db2MonitorSwitches - Get/Update Monitor Switches" in the *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 it with the ACTIVATE DATABASE command, or by maintaining a permanent connection to it.

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
- "System monitor switches" on page 11
- "Database system monitor data organization" on page 4
- "Subsection snapshots" on page 39
- "Global snapshots on partitioned database systems" on page 40

Related tasks:

- "Capturing a database snapshot from a client application" on page 31
- "Capturing a database snapshot from the CLP" on page 27
- "Capturing a database snapshot using SQL" on page 22

Related reference:

- "Snapshot monitor sample output" on page 37

Capturing a database snapshot using SQL

You can capture database snapshots using SQL table functions. Each available snapshot table function corresponds to a snapshot request type.

The 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)

The database manager snapshot SQL table functions are the only exception to this rule as they only have a parameter for partition number. For the database name parameter, if you enter NULL then the monitor uses the database defined by the connection where the table function has been called. For the partition number parameter, enter the integer corresponding to partition number you wish to monitor. To capture a snapshot for the current connected partition, enter a value of -1. To capture a global snapshot, enter a value of -2.

For example, to capture a snapshot of general application information for the current connected partition, and on the database defined by the connection from where this table function call is made:

```
SELECT * FROM TABLE( SNAPSHOT_APPL( cast (NULL as VARCHAR), -1))
                    as SNAPSHOT_APPL
```

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 name for the application id element is appl_id). The following statement will return only the agent id and application id monitor elements:

```
SELECT agent_id, appl_id FROM TABLE(
                        SNAPSHOT_APPL( cast (NULL as VARCHAR), -1))
                        as SNAPSHOT_APPL
```

To obtain a snapshot of a remote instance, you must first attach to that instance.

Prerequisites:

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

Procedure:

1. Optional: Set and check the status of the monitor switches.
2. Capture a snapshot for a particular scope of database activity:
   - *Database manager*:
     - To capture a snapshot of database manager information:

       ```
       SELECT * FROM TABLE( SNAPSHOT_DBM( -1 ))
                       as SNAPSHOT_DBM
       ```

     - To capture a snapshot of database manager information specifically regarding the fast communication manager (FCM):

       ```
       SELECT * FROM TABLE( SNAPSHOT_FCM( -1 ))
                       as SNAPSHOT_FCM
       ```

     - To capture a snapshot of database manager information for a partition specifically regarding the fast communication manager (FCM):

       ```
       SELECT * FROM TABLE( SNAPSHOT_FCMPARTITION( -1 ))
                       as SNAPSHOT_FCMPARTITION
       ```

     - To capture the database manager's monitor switch settings:

       ```
       SELECT * FROM TABLE( SNAPSHOT_SWITCHES( -1 ))
                       as SNAPSHOT_SWITCHES
       ```

   - *Database*: To capture a snapshot of database information:

     ```
     SELECT * FROM TABLE( SNAPSHOT_DATABASE( 'SAMPLE', -1 ))
                     as SNAPSHOT_DATABASE
     ```

   - *Application*:
     - To capture a snapshot of application information:

       ```
       SELECT * FROM TABLE( SNAPSHOT_APPL( 'SAMPLE', -1 ))
                       as SNAPSHOT_APPL
       ```

     - To capture a snapshot of application identification information:

```
SELECT * FROM TABLE( SNAPSHOT_APPL_INFO( 'SAMPLE', -1 ))
               as SNAPSHOT_APPL_INFO
```

– To capture a snapshot of lock wait information:

```
SELECT * FROM TABLE( SNAPSHOT_LOCKWAIT( 'SAMPLE', -1 ))
               as SNAPSHOT_LOCKWAIT
```

– To capture a snapshot of statement information:

```
SELECT * FROM TABLE( SNAPSHOT_STATEMENT( 'SAMPLE', -1 ))
               as SNAPSHOT_STATEMENT
```

– To capture a snapshot of agent information:

```
SELECT * FROM TABLE( SNAPSHOT_AGENT( 'SAMPLE', -1 ))
               as SNAPSHOT_AGENT
```

– To capture a snapshot of subsection information:

```
SELECT * FROM TABLE( SNAPSHOT_SUBSECT( 'SAMPLE', -1 ))
               as SNAPSHOT_SUBSECT
```

- *Buffer pool*: To capture a snapshot of buffer pool information:

```
SELECT * FROM TABLE( SNAPSHOT_BP( 'SAMPLE', -1 ))
              as SNAPSHOT_BP
```

- *Table space*:
  – To capture a snapshot of table space information:

```
SELECT * FROM TABLE( SNAPSHOT_TBS( 'SAMPLE', -1 ))
               as SNAPSHOT_TBS
```

  – To capture a snapshot of table space configuration information:

```
SELECT * FROM TABLE( SNAPSHOT_TBS_CFG( 'SAMPLE', -1 ))
               as SNAPSHOT_TBS_CFG
```

  – To capture a snapshot of table space quiescer information:

```
SELECT * FROM TABLE( SNAPSHOT_QUIESCER( 'SAMPLE', -1 ))
               as SNAPSHOT_QUIESCER
```

  – To capture a snapshot of table space container configuration information:

```
SELECT * FROM TABLE( SNAPSHOT_CONTAINER( 'SAMPLE', -1 ))
               as SNAPSHOT_CONTAINER
```

  – To capture a snapshot of the ranges for a table space map:

```
SELECT * FROM TABLE( SNAPSHOT_RANGES( 'SAMPLE', -1 ))
               as SNAPSHOT_RANGES
```

- *Table*: To capture a snapshot of table information:

```
SELECT * FROM TABLE( SNAPSHOT_TABLE( 'SAMPLE', -1 ))
              as SNAPSHOT_TABLE
```

- *Lock*: To capture a snapshot of lock information:

```
SELECT * FROM TABLE( SNAPSHOT_LOCK( 'SAMPLE', -1 ))
              as SNAPSHOT_LOCK
```

- *Dynamic SQL cache*: To capture a snapshot of dynamic SQL statement cache information:

```
SELECT * FROM TABLE( SNAPSHOT_DYN_SQL( 'SAMPLE', -1 ))
                as SNAPSHOT_DYN_SQL
```

Related concepts:

- "System monitor switches" on page 11
- "Snapshot monitor" on page 21

Related tasks:

- "Setting monitor switches from the CLP" on page 13

Related reference:

- "Snapshot monitor SQL table functions" on page 25
- "Snapshot monitor interface mappings to logical data groups" on page 89

Snapshot monitor SQL table functions

The following table lists all the snapshot table functions. Each table function corresponds to a snapshot request type. 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 2. Snapshot Monitor SQL Table Functions

Monitor level	SQL table function	Information returned
Database manager	SNAPSHOT_DBM	Database manager level information.
Database manager	SNAPSHOT_FCM	Database manager level information regarding the fast communication manager (FCM).
Database manager	SNAPSHOT_FCMPARTITION	Database manager level information for a partition regarding the fast communication manager (FCM).
Database manager	SNAPSHOT_SWITCHES	Database manager monitor switch settings.
Database	SNAPSHOT_DATABASE	Database level information and counters for a database. Information is returned only if there is at least one application connected to the database.
Application	SNAPSHOT_APPL	General 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	SNAPSHOT_APPL_INFO	General application level identification information for each application that is connected to the database on the partition.

Table 2. Snapshot Monitor SQL Table Functions (continued)

Monitor level	SQL table function	Information returned
Application	SNAPSHOT_LOCKWAIT	Application level information regarding lock waits for the applications connected to the database on the partition.
Application	SNAPSHOT_STATEMENT	Application level information regarding statements for the applications connected to the database on the partition. This includes the most recent SQL statement executed (if the statement switch is set).
Application	SNAPSHOT_AGENT	Application level information regarding the agents associated with applications connected to the database on the partition.
Application	SNAPSHOT_SUBSECT	Application level information regarding the subsections of access plans for the applications connected to the database on the partition.
Table	SNAPSHOT_TABLE	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	SNAPSHOT_LOCK	Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.
Table space	SNAPSHOT_TBS	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	SNAPSHOT_TBS_CFG	Information about table space configuration.
Table space	SNAPSHOT_QUIESCER	Information about quiescers at the table space level.
Table space	SNAPSHOT_CONTAINER	Information about table space container configuration at the table space level.
Table space	SNAPSHOT_RANGES	Information about ranges for a table space map.
Buffer pool	SNAPSHOT_BP	Buffer pool activity counters for the specified database. Requires the buffer pool switch.
Dynamic SQL	SNAPSHOT_DYN_SQL	Point–in–time statement information from the SQL statement cache for the database.

Related concepts:

- "Snapshot monitor" on page 21

Related tasks:

- "Capturing a database snapshot using SQL" on page 22

Related reference:

- "Snapshot monitor interface mappings to logical data groups" on page 89

Capturing a database snapshot from the CLP

You can capture database snapshots using the GET SNAPSHOT command from the CLP. 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, or SYSMAINT 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, we will capture a snapshot of 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:

- "System monitor switches" on page 11
- "Snapshot monitor" on page 21
- "Global snapshots on partitioned database systems" on page 40

Related tasks:

- "Setting monitor switches from the CLP" on page 13

Related reference:

- "GET SNAPSHOT Command" in the *Command Reference*
- "Snapshot monitor CLP commands" on page 28
- "Snapshot monitor sample output" on page 37
- "Snapshot monitor interface mappings to logical data groups" on page 89

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 3. Snapshot Monitor CLP Commands

Monitor level	CLP command	Information returned
Connections list	`list applications [show detail]`	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	`list applications for database` *`dbname`* `[show detail]`	Application identification information for each application currently connected to the specified database.
Connections list	`list dcs applications`	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	`get snapshot for dbm`	Database manager level information, including instance-level monitor switch settings.
Database manager	`get dbm monitor switches`	Instance-level monitor switch settings.
Database	`get snapshot for database on` *`dbname`*	Database level information and counters for a database. Information is returned only if there is at least one application connected to the database.
Database	`get snapshot for all databases`	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.

Table 3. Snapshot Monitor CLP Commands (continued)

Monitor level	CLP command	Information returned
Database	`list active databases`	The number of connections to each active database. Includes databases that were started using the ACTIVATE DATABASE command, but have no connections.
Database	`get snapshot for dcs database on dbname`	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	`get snapshot for remote database on dbname`	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	`get snapshot for all remote databases`	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	`get snapshot for application applid appl-id`	Application level information, including cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	`get snapshot for application agentid appl-handle`	Application level information, includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	`get snapshot for applications on dbname`	Application level information for each application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	`get snapshot for all applications`	Application level information for each application that is active on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).
Application	`get snapshot for dcs application applid appl-id`	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 appl-handle`	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 dbname`	Application level information for each DCS application that is connected to the database on the partition. This includes cumulative counters, status information, and most recent SQL statement executed (if statement switch is set).

Table 3. Snapshot Monitor CLP Commands (continued)

Monitor level	CLP command	Information returned
Application	`get snapshot for remote applications on dbname`	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 dbname`	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 appl-id`	List of locks held by the application. Lock wait information requires the lock switch.
Lock	`get snapshot for locks for application agentid appl-handle`	List of locks held by the application. Lock wait information requires the lock switch.
Lock	`get snapshot for locks on dbname`	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 dbname`	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 dbname`	Buffer pool activity counters for the specified database. Requires the buffer pool switch.
Dynamic SQL	`get snapshot for dynamic sql on dbname`	Point-in-time statement information from the SQL statement cache for the database.

Related tasks:

- "Setting monitor switches from the CLP" on page 13
- "Capturing a database snapshot from the CLP" on page 27

Related reference:

- "GET SNAPSHOT Command" in the *Command Reference*
- "Snapshot monitor interface mappings to logical data groups" on page 89

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, or SYSMAINT 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.

   ```
   #define SNAPSHOT_BUFFER_UNIT_SZ 102400
   ```

4. Declare the sqlca, sqlma, db2GetSnapshotData, and sqlm_collected structures. Also, initialize a pointer to contain the snapshot buffer, and establish the buffer's size.

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

   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',
           sizeof(struct pRequestedDataGroups));
   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', 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_DEFAULT;
```

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

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

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

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

10. Process the snapshot monitor data stream.
11. Clear the buffer.

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

Related concepts:

- "System monitor switches" on page 11
- "Snapshot monitor" on page 21
- "System monitor output: the self-describing data stream" on page 6
- "Snapshot monitor self-describing data stream" on page 41

Related tasks:

- "Setting monitor switches from a client application" on page 16

Related reference:

- "db2GetSnapshot - Get Snapshot" in the *Administrative API Reference*
- "db2MonitorSwitches - Get/Update Monitor Switches" in the *Administrative API Reference*
- "db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot Output Buffer" in the *Administrative API Reference*
- "db2ResetMonitor - Reset Monitor" in the *Administrative API Reference*
- "Snapshot monitor API request types" on page 33

Related samples:

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

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 4. 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.
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).

Table 4. Snapshot Monitor API Request Types (continued)

Monitor level	API request type	Information returned
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.

Table 4. Snapshot Monitor API Request Types (continued)

Monitor level	API request type	Information returned
Lock	SQLMA_APPL_LOCKS	List of locks held by the application. Lock wait information requires the lock switch.
Lock	SQLMA_APPL_LOCKS_AGENT_ID	List of locks held by the application. Lock wait information requires the lock switch.
Lock	SQLMA_DBASE_LOCKS	Lock information at the database level, and application level for each application connected to the database. Requires the lock switch.
Table space	SQLMA_DBASE_TABLESPACES	Information about table space activity at the database level, the application level for each application connected to the database, and the table space level for each table space that has been accessed by an application connected to the database. Requires the buffer pool switch.
Buffer pool	SQLMA_BUFFERPOOLS_ALL	Buffer pool activity counters. Requires the buffer pool switch.
Buffer pool	SQLMA_DBASE_BUFFERPOOLS	Buffer pool activity counters for the specified database. Requires the buffer pool switch.
Dynamic SQL	SQLMA_DYNAMIC_SQL	Point-in-time statement information from the SQL statement cache for the database.

Related concepts:

- "Snapshot monitor" on page 21
- "Snapshot monitor self-describing data stream" on page 41

Related tasks:

- "Setting monitor switches from a client application" on page 16
- "Capturing a database snapshot from a client application" on page 31

Related reference:

- "db2GetSnapshot - Get Snapshot" in the *Administrative API Reference*
- "db2MonitorSwitches - Get/Update Monitor Switches" in the *Administrative API Reference*
- "db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot Output Buffer" in the *Administrative API Reference*
- "db2ResetMonitor - Reset Monitor" in the *Administrative API Reference*
- "db2ConvMonStream - Convert Monitor Stream" in the *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
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 V Lock
Mode                        = S

Lock Name                   = 0x41414141414A48520000000041
Lock Attributes             = 0x00000000
Release Flags               = 0x40000000
Lock Count                  = 1
Hold Count                  = 0
Lock Object Name            = 0
Object Type                 = Internal P 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 P 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:

- "System monitor switches" on page 11
- "Snapshot monitor" on page 21

Related tasks:

- "Setting monitor switches from the CLP" on page 13

Related reference:

- "Snapshot monitor CLP commands" on page 28

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:

- "System monitor switches" on page 11
- "Snapshot monitor" on page 21
- "Global snapshots on partitioned database systems" on page 40

Related reference:

- "Snapshot monitor CLP commands" on page 28

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**

  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 5
- "Subsection snapshots" on page 39
- "Snapshot monitor" on page 21

Related tasks:

- "Capturing a database snapshot from a client application" on page 31
- "Capturing a database snapshot from the CLP" on page 27
- "Capturing a database snapshot using SQL" on page 22

Snapshot monitor self-describing data stream

After you capture a snapshot with the db2GetSnapshot API, it returns the snapshot output as a self-describing data stream. Figure 2 on page 42 shows the structure of the data stream and Table 5 on page 43 provides some examples of the logical data groups and monitor elements that may 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, collected would appear as SQLM_ELM_COLLECTED in the snapshot monitor output. Types are prefixed with **SQLM_TYPE_** in the actual data stream. For example, headers appear as SQLM_TYPE_HEADER in the data stream.

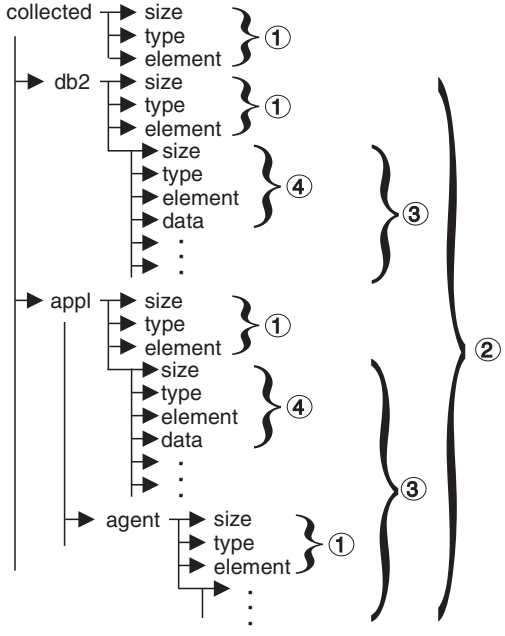


Figure 2. Snapshot Monitor Data Stream

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

Table 5. Sample Snapshot Data Stream

Logical Data Group	Data Stream	Description
collected	1000 header collected	Size of snapshot data (in bytes). Indicates the start of a logical data group. Name of the logical data group.
	4 u32bit server_db2_type sqlf_nt_server	Size of the data stored in this monitor element. Monitor element type - unsigned 32 bit numeric. The name of the monitor element collected. The collected value for this element.
	2 u16bit node_number 3	Size of the data stored in this monitor element. Monitor element type - unsigned 16 bit numeric. The name of the monitor element collected. The collected value for this element.
db2	200 header db2	Size of the db2 level portion of data in the snapshot. Indicates the start of a logical data group. Name of the logical data group.
	4 u32bit sort_heap_allocated 16	Size of the data stored in this monitor element. Monitor element type - unsigned 32 bit numeric. The name of the monitor element collected. The collected value for this element.
	4 u32bit local_cons 3	Size of the data stored in this monitor element. Monitor element type - unsigned 32 bit numeric. The name of the monitor element collected. The collected value for this element.
	. . .	. . .
appl	100 header appl	Size of the appl element data in the snapshot. Indicates the start of a logical data group. Name of the logical data group.
	4 u32bit locks_held 3	Size of the data stored in this monitor element. Monitor element type - unsigned 32 bit numeric. The name of the monitor element collected. The collected value for this element.
	. . .	. . .
agent	50 header agent	Size of the agent portion of the appl structure. Indicates the start of a logical data group. Name of the logical data group.
	4 u32bit agent_pid 12	Size of the data stored in this monitor element. Monitor element type - 32 bit numeric. The name of the monitor element collected. 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 31

Related reference:

- "Snapshot monitor API request types" on page 33
- "Snapshot monitor interface mappings to logical data groups" on page 89

Related samples:

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

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: for instance, connections, deadlocks, statements, and 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. 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 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 47

Event monitors

- "Creating an event monitor" on page 50

Related reference:

- "Event monitor sample output" on page 71
- "Event types" on page 46

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 6. 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.
STATEMENTS	End of SQL statement	Statement start/stop time, CPU used, text of dynamic SQL, SQLCA (return code of SQL statement), and other metrics such as fetch count.
	End of subsection	For partitioned databases: CPU consumed, execution time, table and tablequeue information.
TRANSACTIONS	End of unit of work	UOW work start/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.
DATABASE	Database deactivation	All database level counters.

Table 6. Event Types (continued)

Event type	When data is collected	Available information
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/written for each table.

Note: A detailed deadlock event monitor is created for each newly created database. This event monitor, named DFTDLOCKMON, starts when the database is activated and will write to files in the database directory. You can avoid the overhead incurred by this event monitor by dropping it.

Related concepts:

- "Event monitors" on page 45
- "Counter status and visibility" on page 5
- "Event type mappings to logical data groups" on page 132

Related tasks:

- "Creating an event monitor" on page 50

Related reference:

- "Event monitor sample output" on page 71

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. See the Related task, ″Creating an event monitor″.
2. *For file and pipe event monitors only*: Ensure that the directory or named pipe that will receive the event records exists. The event monitor will not activate otherwise.

   On AIX, you can create named pipes by using the mkfifo command. On Linux and other UNIX types (such as Solaris) use the pipe() routine.

   On Windows NT/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 NT/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;
   ```

   When started, an 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-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 NT/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 NT/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 45
- "Event records and their corresponding applications" on page 80

Related tasks:

- "Creating an event monitor" on page 50
- "Creating an event monitor for partitioned databases" on page 67
- "Formatting file or pipe event monitor output from a command line" on page 70

Related reference:

- "Event monitor sample output" on page 71

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. 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 62
- "Event monitor named pipe management" on page 66
- "Event monitors" on page 45
- "Event monitor table management" on page 54

Related tasks:

- "Creating a table event monitor" on page 51
- "Creating a file event monitor" on page 59
- "Creating a pipe event monitor" on page 64
- "Creating an event monitor for partitioned databases" on page 67

Related reference:

- "CREATE EVENT MONITOR statement" in the *SQL Reference, Volume 2*
- "Event monitor sample output" on page 71
- "Event types" on page 46

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.

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 dlmon FOR eventtype
                       WRITE TO TABLE
   ```

   `dlmon` is the name of the event monitor.
2. Specify the types of events to be monitored. You can monitor one or more event types with a single event monitor.

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

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

```
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 monitors" on page 45
- "Event monitor table management" on page 54
- "Event records and their corresponding applications" on page 80

Related tasks:

- "Collecting information about database system events" on page 47

Related reference:

- "Event types" on page 46

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. Monitor elements of type sqlm_time (elapsed time) are an exception. The column names for such types are type_name_s, and type_name_ms, representing the columns that store the time in seconds and microseconds, respectively. Any event monitor element that does not have a corresponding target table column is ignored.

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

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

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

You can specify the schema, table, and table space names in the CREATE EVENT MONITOR statement. The schema name is provided along with the table name, forming a derived name for the table. A target table can only be used by a single event monitor. If a target table is found to already be defined for another event monitor, or if it cannot be created for any other reason, the CREATE EVENT MONITOR statement will fail. The table space name can be added after the table name with the optional IN clause. Unlike the target tables, which DB2® creates, a table space must exist if it is included in an event monitor definition. 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.

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 7. Write-to-Table Event Monitor Target Tables

Event type	Target table names	Available information
DEADLOCKS	CONNHEADER	Connection metadata
	DEADLOCK	Deadlock data
	DLCONN	Applications and locks involved in deadlock
	CONTROL	Event monitor metadata
DEADLOCKS WITH DETAILS	CONNHEADER	Connection metadata
	DEADLOCK	Deadlock data
	DLCONN	Applications involved in deadlock
	DLLOCK	Locks involved in deadlock
	CONTROL	Event monitor metadata
STATEMENTS	CONNHEADER	Connection metadata
	STMT	Statement data
	SUBSECTION	Statement data specific to subsection
	CONTROL	Event monitor metadata
TRANSACTIONS	CONNHEADER	Connection metadata
	XACT	Transaction data
	CONTROL	Event monitor metadata
CONNECTIONS	CONNHEADER	Connection metadata
	CONN	Connection data
	CONTROL	Event monitor metadata
DATABASE	DB	Database manager data
	CONTROL	Event monitor metadata
BUFFERPOOLS	BUFFERPOOL	Buffer pool data
	CONTROL	Event monitor metadata
TABLESPACES	TABLESPACE	Tablespace data
	CONTROL	Event monitor metadata
TABLES	TABLE	Table data
	CONTROL	Event monitor metadata

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 8. System Monitor Data Type Mappings

System monitor data type	SQL data type
SQLM_TYPE_STRING	CHAR[n], VARCHAR[n], CLOB[n]
SQLM_TYPE_U8BIT and SQLM_TYPE_8BIT	SMALLINT, INTEGER, or BIGINT
SQLM_TYPE_U16BIT and SQLM_TYPE_16BIT	SMALLINT, INTEGER, or BIGINT
SQLM_TYPE_U32BIT and SQLM_TYPE_32BIT	INTEGER or BIGINT
SQLM_TYPE_U64BIT and SQLM_TYPE_64BIT	BIGINT
SQLM_TIMESTAMP	TIMESTAMP
SQLM_TIME	2 columns of type INTEGER (one for seconds: element_name_s, another for milliseconds: element_name_ms)
SQLCA: SQLERRMC	VARCHAR[72]
SQLCA: SQLSTATE	CHAR[5]
SQLCA: SQLWARN	CHAR[11]
SQLCA: other fields	INTEGER or BIGINT

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).

Related concepts:

- "Event monitors" on page 45
- "Event type mappings to logical data groups" on page 132
- "Write-to-table and file event monitor buffering" on page 63

Related tasks:

- "Creating a table event monitor" on page 51

Related reference:

- "Event Monitor Name" on page 424

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

   `dlmon` is the name of the event monitor.

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

2. Specify the types of events to be monitored. You can monitor one or more event types with a single event monitor.

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

   This event monitor will monitor for the CONNECTIONS and DEADLOCKS WITH DETAILS event types.

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

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

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

   The default (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 dlmon FOR CONNECTIONS, DEADLOCKS WITH DETAILS
                       WRITE TO FILE '/tmp/dlevents' BUFFERSIZE 8
                       BLOCKED
   ```

   Event monitors are blocked by default.

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

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

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

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

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

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

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

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

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

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

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

```
CREATE EVENT MONITOR 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 62
- "Event monitors" on page 45
- "Event type mappings to logical data groups" on page 132
- "Write-to-table and file event monitor buffering" on page 63

Related tasks:

- "Collecting information about database system events" on page 47
- "Creating an event monitor for partitioned databases" on page 67
- "Formatting file or pipe event monitor output from a command line" on page 70

Related reference:

- "Event monitor sample output" on page 71
- "Event types" on page 46

Event monitor file management

A file event monitor enables the event monitor to store its event records in files. All the output of the event monitor goes in the directory supplied in the FILE parameter for the CREATE EVENT MONITOR statement. This directory will not be created by DB2® if it does not exist. Before the monitor is activated, the directory must exist, or the SET EVENT MONITOR command will return an error. When a file event monitor is first activated, a control file named db2event.ctl is created in this directory. Do not remove or modify this file.

By default, an event monitor writes its trace to a single file, called 00000000.evt. This file keeps growing as long as there is space on the file system. If you specified a file size limit with the MAXFILESIZE parameter of the CREATE EVENT MONITOR statement, then when a file is full, output is automatically directed to the next file. Hence, the active file is the file with the highest number.

You can limit the maximum size of the entire event monitor trace by also using the MAXFILES parameter of the CREATE EVENT MONITOR statement. When the number of files reaches the maximum defined by MAXFILES, the event monitor deactivates itself and the following message is written to the administration notification log.

```
DIA1601I Event Monitor monitor-name was deactivated when it reached
its preset MAXFILES and MAXFILESIZE limit.
```

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

If a file event monitor runs out of disk space, it shuts itself down after logging a system-error-level message in the administration notification log.

When a file event monitor is restarted, it can either erase any existing data or append new data to it. This option is specified in the CREATE EVENT MONITOR statement, where either an APPEND monitor or a REPLACE monitor can be created. APPEND is the default option. An APPEND event monitor starts writing at the end of the file it was last using. If you have removed that file, the next file number in sequence is used. When an append event monitor is restarted, only a start_event is generated. The event log header and database header are generated only for the first activation. A REPLACE event monitor always deletes existing event files and starts writing at 00000000.evt.

You may want to process monitor data while the event monitor is active. This is possible, and furthermore, when you are finished processing a file, you can delete it, freeing up space for further monitoring data. An event monitor

cannot be forced to switch to the next file unless you stop and restart it. It must also be in APPEND mode. In order to keep track of which events have been processed in the active file, you can create an application that simply keeps track of the file number and location of the last record processed. When processing the trace the next time around, the application can simply seek to that file location.

Related concepts:

- "Event monitors" on page 45
- "Event type mappings to logical data groups" on page 132
- "Write-to-table and file event monitor buffering" on page 63
- "System monitor output: the self-describing data stream" on page 6
- "Event monitor self-describing data stream" on page 81

Related tasks:

- "Collecting information about database system events" on page 47
- "Creating a file event monitor" on page 59

Related reference:

- "Event monitor sample output" on page 71

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 incurring 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 db2diag.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 the entirety 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 62
- "Event monitors" on page 45
- "Event monitor table management" on page 54

Related tasks:

- "Creating a file event monitor" on page 59
- "Creating a table event monitor" on page 51

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.

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 66
- "System monitor output: the self-describing data stream" on page 6
- "Event monitor self-describing data stream" on page 81

Related reference:

- "Event monitor sample output" on page 71

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 Solaris) use the pipe() routine. On Windows NT/2000, 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 monitors" on page 45
- "Event type mappings to logical data groups" on page 132
- "System monitor output: the self-describing data stream" on page 6
- "Event monitor self-describing data stream" on page 81

Related tasks:

- "Collecting information about database system events" on page 47
- "Creating an event monitor" on page 50

Related reference:

- "Event monitor sample output" on page 71

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 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.

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

The first column of each target table is named PARTITION_KEY, and is used as the partitioning key for the table. The value of this column is chosen so that each event monitor process inserts data into the database partition on which the process is running; that is, insert operations are performed locally on the database partition where the event monitor process is running. On any database partition, the PARTITION_KEY field will contain the same value. This means that if a data partition is dropped and data redistribution is performed, all data on the dropped database partition will go to one other

database partition instead of being evenly distributed. Therefore, before removing a database partition, consider deleting all table rows on that database partition.

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

The table space within which target tables are defined must exist across all partitions that will have event monitor data written to them. Failure to observe this rule will result in records not being written to the log on partitions (with event monitors) where the table space does not exist. Events will still be written on partitions where the table space does exist, and no error will be returned. This behavior allows users to choose a subset of partitions for monitoring, by creating a table space that exists only on certain partitions.

During write-to-table event monitor activation, the CONTROL table rows for FIRST_CONNECT and EVMON_START are only inserted on the catalog database partition. This requires that the table space for the control table exist on the catalog database partition. If it does not exist on the catalog database partition, these inserts are not performed. If a partition is not yet active when a write-to-table event monitor is activated, 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 dlmon FOR DEADLOCKS
                     WRITE TO FILE '/tmp/dlevents'
                     ON PARTITION 3
```

   `dlmon` represents the name of the event monitor.

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

3 represents the partition number to be monitored.

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

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

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

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

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

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

3. 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:

- "Event monitors" on page 45
- "Counter status and visibility" on page 5
- "Event type mappings to logical data groups" on page 132

Related tasks:

- "Creating a file event monitor" on page 59
- "Creating a table event monitor" on page 51

Related reference:

- "Event monitor sample output" on page 71
- "Event types" on page 46

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).

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
- SYSMAINT
- 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 62
- "Event monitor named pipe management" on page 66
- "Event monitor self-describing data stream" on page 81

Related tasks:

- "Creating a file event monitor" on page 59
- "Creating a pipe event monitor" on page 64

Event monitor sample output

To illustrate the nature of event monitoring, here is a 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/db2sampl`
- Windows 2000/NT: `sqllib\bin\db2sampl.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.

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:\dlmon
```

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

Monitor:

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

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: SQL08010
  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: E:\DB2\NODE0000\SQL00001\
  First connection timestamp: 06-03-2002 12:20:21.713144
  Event Monitor Start time:   06-03-2002 12:28:29.296150
------------------------------------------------------------------------
```

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, we see deadlock events, spawned by the deadlock state incurred by the previous statements.

```
3) Deadlock Event ...
  Deadlock ID:   1
  Number of applications deadlocked: 2
  Deadlock detection time: 06-03-2002 12:31:32.812477
  Rolled back Appl participant no: 2
  Rolled back Appl Id: *LOCAL.DB2.001703163033
  Rolled back Appl seq number: : 0001

4) Connection Header Event ...
  Appl Handle: 11
```

```
  Appl Id: *LOCAL.DB2.001703163033
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.DB2.001703163033
  Program Name    : db2bp.exe
  Authorization Id: ADMINISTRATOR
  Execution Id    : ADMINISTRATOR
  Codepage Id: 1252
  Territory code: 1
  Client Process Id: 738280448
  Client Database Alias: SAMPLE
  Client Product Id: SQL08010
  Client Platform: Unknown
  Client Communication Protocol: Local
  Client Network Name:
  Connect timestamp: 06-03-2002 12:30:32.842438

5) Deadlocked Connection ...
  Deadlock ID:   1
  Participant no.: 2
  Participant no. holding the lock: 1
  Appl Id: *LOCAL.DB2.001703163033
  Appl Seq number: 0001
  Appl Id of connection holding the lock: *LOCAL.DB2.001703163033
  Seq. no. of connection holding the lock: 0001
  Lock wait start time:
  Lock Name       : 0x02000300270000000000000052
  Lock Attributes : 0x00000000
  Release Flags   : 0x00000001
  Lock Count      : 1
  Hold Count      : 0
  Current Mode    : none
  Deadlock detection time: 06-03-2002 12:31:32.812602
  Table of lock waited on      : STAFF
  Schema of lock waited on     : ADMINISTRATOR

  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 occured on: 0
  Lock object name: 39
  Application Handle: 11
  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                   : 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               : ADMINISTRATOR

Table Name                 : STAFF
Mode                       : IS  - Intent Share

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                 : 255
Hold Count                 : 0
Lock Object Name           : 13
Object Type                : Row
Tablespace Name            : USERSPACE1
Table Schema               : ADMINISTRATOR

Table Name                 : DEPARTMENT
Mode                       : X   - Exclusive

Lock Name                  : 0x020004000000000000000000054
Lock Attributes            : 0x00000000
Release Flags              : 0x40000000
Lock Count                 : 255
Hold Count                 : 0
Lock Object Name           : 4
Object Type                : Table
Tablespace Name            : USERSPACE1
Table Schema               : ADMINISTRATOR

Table Name                 : DEPARTMENT
Mode                       : IX  - Intent Exclusive

Lock Name                  : 0x41414141414A48520000000041
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                  : 0x434F4E544F4B4E310000000041
```

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

  Locks Held:    6
  Locks in List: 6

6) Connection Header Event ...
  Appl Handle: 10
  Appl Id: *LOCAL.DB2.001C83162950
  Appl Seq number: 0001
  DRDA AS Correlation Token: *LOCAL.DB2.001C83162950
  Program Name    : db2bp.exe
  Authorization Id: ADMINISTRATOR
  Execution Id    : ADMINISTRATOR
  Codepage Id: 1252
  Territory code: 1
  Client Process Id: 3909148416
  Client Database Alias: SAMPLE
  Client Product Id: SQL08010
  Client Platform: Unknown
  Client Communication Protocol: Local
  Client Network Name:
  Connect timestamp: 06-03-2002 12:29:50.171226

7) Deadlocked Connection ...
  Deadlock ID:   1
  Participant no.: 1
  Participant no. holding the lock: 2
  Appl Id: *LOCAL.DB2.001C83162950
  Appl Seq number: 0001
  Appl Id of connection holding the lock: *LOCAL.DB2.001C83162950
  Seq. no. of connection holding the lock: 0001

  Lock wait start time:
  Lock Name       : 0x020004000D0000000000000052
  Lock Attributes : 0x00000000
  Release Flags   : 0x00000001
  Lock Count      : 1
  Hold Count      : 0
  Current Mode    : none
  Deadlock detection time: 06-03-2002 12:31:32.832898
  Table of lock waited on      : DEPARTMENT
  Schema of lock waited on     : ADMINISTRATOR

  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 occured on: 0
  Lock object name: 13
```

```
Application Handle: 10
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               : ADMINISTRATOR

    Table Name                 : DEPARTMENT
    Mode                       : NS  - Share (and Next Key Share)

    Lock Name                  : 0x02000400000000000000000054
    Lock Attributes            : 0x00000000
    Release Flags              : 0x00000001
    Lock Count                 : 1
    Hold Count                 : 0
    Lock Object Name           : 4
    Object Type                : Table
    Tablespace Name            : USERSPACE1
    Table Schema               : ADMINISTRATOR

    Table Name                 : DEPARTMENT
    Mode                       : IS  - Intent Share

    Lock Name                  : 0x0100000001000000100640056
    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                  : 0x020003002700000000000000052
    Lock Attributes            : 0x00000000
    Release Flags              : 0x40000000
    Lock Count                 : 255
    Hold Count                 : 0
    Lock Object Name           : 39
    Object Type                : Row
    Tablespace Name            : USERSPACE1
    Table Schema               : ADMINISTRATOR
```

```
      Table Name                 : STAFF
      Mode                       : X   - Exclusive

      Lock Name                  : 0x020003000000000000000000054
      Lock Attributes            : 0x00000000
      Release Flags              : 0x40000000
      Lock Count                 : 255
      Hold Count                 : 0
      Lock Object Name           : 3
      Object Type                : Table
      Tablespace Name            : USERSPACE1
      Table Schema               : ADMINISTRATOR

      Table Name                 : STAFF
      Mode                       : IX  - Intent Exclusive

      Lock Name                  : 0x41414141414A48520000000041
      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                  : 0x434F4E544F4B4E310000000041
      Lock Attributes            : 0x00000000
      Release Flags              : 0x40000000
      Lock Count                 : 1
      Hold Count                 : 0
      Lock Object Name           : 0
      Object Type                : Internal - Plan
      Mode                       : S   - Share

  Locks Held:    7
  Locks in List: 7
```

Epilog:

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

```
8) Table Event ...
  Table schema: ADMINISTRATOR
  Table name: DEPARTMENT

  Record is the result of a flush: FALSE
  Table type: User
  Rows read: 9
  Rows written: 1
  Overflow Accesses: 0
  Page reorgs: 0
  Table event timestamp: 06-03-2002 12:32:07.396356
```

```
9) Table Event ...
  Table schema: ADMINISTRATOR
  Table name: STAFF

  Record is the result of a flush: FALSE
  Table type: User
  Rows read: 35
  Rows written: 1
  Overflow Accesses: 0
  Page reorgs: 0
  Table event timestamp: 06-03-2002 12:32:07.396393

10) Table Event ...
  Table schema: SYSIBM
  Table name: SYSTABLES

  Record is the result of a flush: FALSE
  Table type: Catalog
  Rows read: 2
  Rows written: 0
  Overflow Accesses: 0
  Page reorgs: 0
  Table event timestamp: 06-03-2002 12:32:07.396406
```

Related concepts:

- "Event monitors" on page 45
- "Event type mappings to logical data groups" on page 132

Related tasks:

- "Collecting information about database system events" on page 47

Related reference:

- "Event types" on page 46

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 monitors" on page 45
- "Event monitor self-describing data stream" on page 81

Related tasks:

- "Collecting information about database system events" on page 47

Related reference:

- "Event monitor sample output" on page 71

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 on page 82 shows the structure of the data stream and Table 9 on page 83 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.

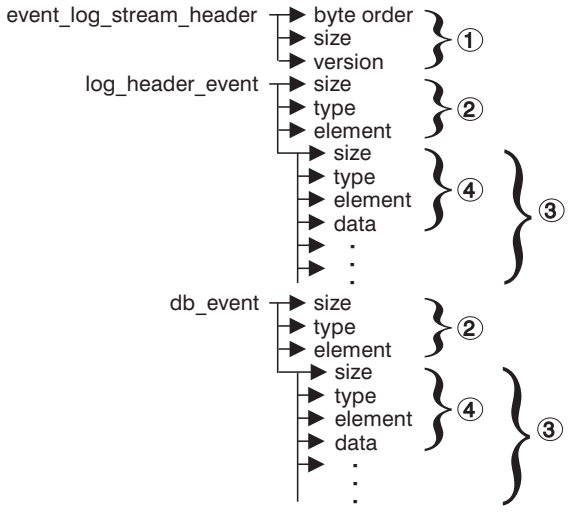


Figure 3. Event Monitor Data Stream

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

Table 9. Sample event data stream

Logical Data Group	Data Stream	Description
event_log_stream_header	sqlm_little_endian 200 sqlm_dbmon_version8	Not used (for compatibility with previous releases). Not used (for compatibility with previous releases). 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 header log_header	Size of the logical data group. Indicates the start of a logical data group. Name of the logical data group.
	4 u32bit byte_order little_endian	Size of the data stored in this monitor element. Monitor element type - 32 bit numeric. The name of the monitor element collected. The collected value for this element.
	2 u16bit codepage_id 850	Size of the data stored in this monitor element. Monitor element type - unsigned 16 bit numeric. The name of the monitor element collected. The collected value for this element.
db_event	100 header db_event	Size of the logical data group. Indicates the start of a logical data group. Name of the logical data group.
	4 u32bit lock_waits 2	Size of the data stored in this monitor element. Monitor element type - unsigned 32 bit numeric. The name of the monitor element collected. 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 may 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, Windows® NT/2000 to UNIX). Codepage translation may 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 62
- "Event monitor named pipe management" on page 66
- "Event type mappings to logical data groups" on page 132

Related tasks:

- "Transferring event monitor data between systems" on page 84

Related reference:

- "Event monitor sample output" on page 71

Related samples:

- "bldevm -- Builds event monitor program, evm, on AIX"
- "bldevm.bat -- Builds event monitor program, evm, on Windows"
- "evm.c -- Process event monitor data on AIX (C)"
- "evm.c -- Process event monitor data on Windows (C)"
- "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)"
- "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(l) ((((l) >> 24) & 0xFF) | ((((l) & 0xFF0000) >> 8) & 0xFF00) \
                    | (((l) & 0xFF00) << 8) | ((l) << 24))
```

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

int HeaderByteReverse( sqlm_header_info * pHeader)
{   int rc = 0;
    pHeader->size = SWAP4(pHeader->size);
    pHeader->type = SWAP2(pHeader->type);
    pHeader->element = SWAP2(pHeader->element);

    return rc;
}

int DataByteReverse( char * dataBuf, sqluint32 dataSize)
{   int rc = 0;
    sqlm_header_info * pElemHeader = NULL;
    char * pElemData = NULL;
    sqluint32 dataOffset = 0;
    sqluint32 elemDataSize  = 0;
    sqluint32 elemHeaderSize = sizeof( sqlm_header_info);

    // For each of the elements in the datastream 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 62
- "Event monitor named pipe management" on page 66
- "Event monitor self-describing data stream" on page 81

Part 2. System Monitor Reference

Chapter 5. Logical Data Groups

Snapshot monitor interface mappings to logical data groups

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 10. Snapshot Monitor Interface Mappings to Logical Data Groups

API request type	CLP command	SQL table function	Logical data groups
SQLMA_APPLINFO_ALL	list applications [show detail]		appl_info
SQLMA_DBASE_APPLINFO	list applications for database *dbname* [show detail]		appl_info
SQLMA_DCS_APPLINFO_ALL	list dcs applications [show detail]		dcs_appl_info
SQLMA_DB2	get snapshot for dbm	SNAPSHOT_DBM	db2
		SNAPSHOT_FCM	fcm
		SNAPSHOT_FCMNODE	fcm_node
			memory_pool
	get dbm monitor switches	SNAPSHOT_SWITCHES	switch_list
SQLMA_DBASE	get snapshot for database on *dbname*	SNAPSHOT_DATABASE	dbase
			rollforward, tablespace, memory_pool
SQLMA_DBASE_ALL	get snapshot for all databases		dbase, rollforward, tablespace, memory_pool
	list active databases		dbase
SQLMA_DCS_DBASE	get snapshot for dcs database on *dbname*		dcs_dbase, stmt_transmissions
SQLMA_DCS_DBASE_ALL	get snapshot for all databases		dcs_dbase, stmt_transmissions
SQLMA_DBASE_REMOTE	get snapshot for remote database on *dbname*		dbase_remote
SQLMA_DBASE_REMOTE_ALL	get snapshot for all remote databases		dbase_remote

Table 10. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

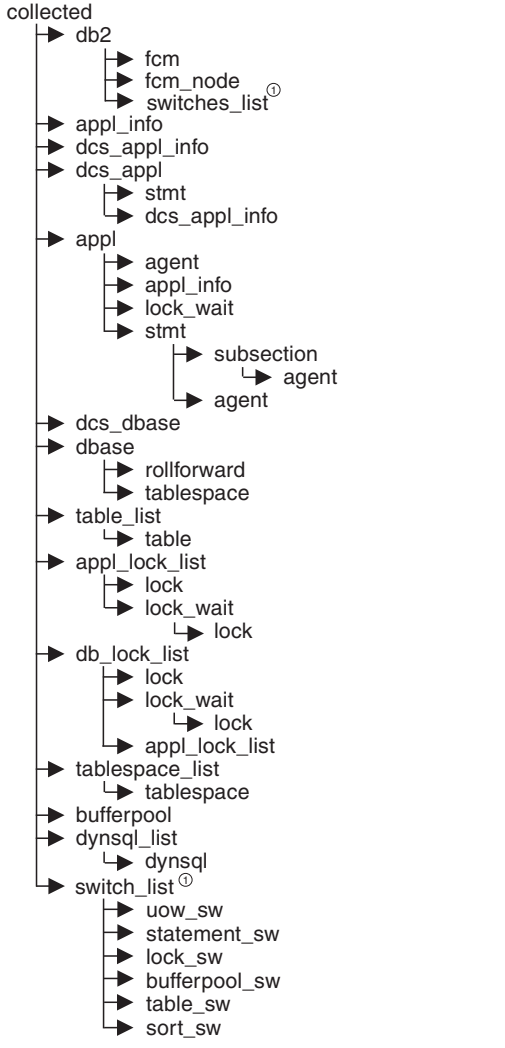
API request type	CLP command	SQL table function	Logical data groups
SQLMA_APPL	get snapshot for application applid *appl-id*		appl, agent, appl_info, lock_wait, stmt, subsection, memory_pool
SQLMA_AGENT_ID	get snapshot for application agentid *appl-handle*		appl, agent, appl_info, lock_wait, stmt, subsection, memory_pool
SQLMA_DBASE_APPLS	get snapshot for applications on *dbname*	SNAPSHOT_APPL	appl, agent
		SNAPSHOT_APPL_INFO	appl_info
		SNAPSHOT_LOCKWAIT	appl, lock_wait
		SNAPSHOT_STATEMENT	appl, stmt
		SNAPSHOT_AGENT	appl, agent, stmt
		SNAPSHOT_SUBSECT	appl, subsection, stmt
			memory_pool
SQLMA_APPL_ALL	get snapshot for all applications	SNAPSHOT_APPL	appl, agent
		SNAPSHOT_APPL_INFO	appl_info
		SNAPSHOT_LOCKWAIT	appl, lock_wait
		SNAPSHOT_STATEMENT	appl, stmt
		SNAPSHOT_AGENT	appl, agent, stmt
		SNAPSHOT_SUBSECT	appl, subsection, stmt
			memory_pool
SQLMA_DCS_APPL	get snapshot for dcs application applid *appl-id*		dcs_appl, dcs_stmt, dcs_appl_info, stmt_transmissions
SQLMA_DCS_APPL_ALL	get snapshot for all dcs applications		dcs_appl, dcs_stmt, dcs_appl_info, stmt_transmissions
SQLMA_DCS_APPL_HANDLE	get snapshot for dcs application agentid *appl-handle*		dcs_appl, dcs_stmt, dcs_appl_info, stmt_transmissions
SQLMA_DCS_DBASE_APPLS	get snapshot for dcs applications on *dbname*		dcs_appl, dcs_stmt, dcs_appl_info, stmt_transmissions
SQLMA_DBASE_APPLS_REMOTE	get snapshot for remote applications on *dbname*		dbase_appl
SQLMA_APPL_REMOTE_ALL	get snapshot for all remote applications		dbase_appl
SQLMA_DBASE_TABLES	get snapshot for tables on *dbname*	SNAPSHOT_TABLE	table
			table_list, table_reorg
SQLMA_APPL_LOCKS	get snapshot for locks for application applid *appl-id*		appl_lock_list, lock_wait, lock

Table 10. Snapshot Monitor Interface Mappings to Logical Data Groups (continued)

API request type	CLP command	SQL table function	Logical data groups
SQLMA_APPL_LOCKS_AGENT_ID	get snapshot for locks for application agentid *appl-handle*		appl_lock_list, lock_wait, lock
SQLMA_DBASE_LOCKS	get snapshot for locks on *dbname*	SNAPSHOT_LOCK	appl_lock_list, lock
			db_lock_list, lock_wait
SQLMA_DBASE_TABLESPACES	get snapshot for tablespaces on *dbname*	SNAPSHOT_TBS	tablespace
		SNAPSHOT_TBS_CFG	tablespace, tablespace_nodeinfo
		SNAPSHOT_QUIESCER	tablespace_quiescer, tablespace_nodeinfo
		SNAPSHOT_CONTAINER	tablespace_container, tablespace_nodeinfo
		SNAPSHOT_RANGES	tablespace_ranges, tablespace_nodeinfo
			tablespace_list, tablespace_nodeinfo
SQLMA_BUFFERPOOLS_ALL	get snapshot for all bufferpools	SNAPSHOT_BP	bufferpool
SQLMA_DBASE_BUFFERPOOLS	get snapshot for bufferpools on *dbname*	SNAPSHOT_BP	bufferpool
SQLMA_DYNAMIC_SQL	get snapshot for dynamic sql on *dbname*	SNAPSHOT_DYN_SQL	dynsql
			dynsql_list

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

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.

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 11. Snapshot Monitor Logical Data Groups and Monitor Elements

Snapshot logical data groups	Monitor element name	Monitor element title
collected	**node_number**	"Node Number" on page 190
	server_db2_type	"Database Manager Type at Monitored (Server) Node" on page 158
	server_instance_name	"Server Instance Name" on page 157
	server_nname	"Configuration NNAME at Monitoring (Server) Node" on page 157
	server_prdid	"Server Product/Version ID" on page 159
	server_switch_list	Monitor switches control data
	server_version	"Server Version" on page 159
	time_stamp	"Snapshot Time" on page 420
	time_zone_disp	"Time Zone Displacement" on page 162
db2	**agents_registered**	"Agents Registered" on page 205
	agents_waiting_on_token	"Agents Waiting for a Token" on page 206
	con_local_dbases	"Local Databases with Current Connects" on page 203
	local_cons	"Local Connections" on page 202
	local_cons_in_exec	"Local Connections Executing in the Database Manager" on page 202
	piped_sorts_requested	"Piped Sorts Requested" on page 219
	piped_sorts_accepted	"Piped Sorts Accepted" on page 220
	post_threshold_sorts	"Post Threshold Sorts" on page 219
	produce_name	"Product Name" on page 161
	rem_cons_in	"Remote Connections To Database Manager" on page 200
	rem_cons_in_exec	"Remote Connections Executing in the Database Manager" on page 201
	service_level	"Service Level" on page 160
	sort_heap_allocated	"Total Sort Heap Allocated" on page 218

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
db2 (continued)	**agents_created_empty_pool**	"Agents Created Due to Empty Agent Pool" on page 209
	agents_from_pool	"Agents Assigned From Pool" on page 208
	agents_registered_top	"Maximum Number of Agents Registered" on page 206
	agents_stolen	"Stolen Agents" on page 210
	agents_waiting_top	"Maximum Number of Agents Waiting" on page 207
	comm_private_mem	"Committed Private Memory" on page 211
	coord_agents_top	"Maximum Number of Coordinating Agents" on page 209
	db2start_time	"Start Database Manager Timestamp" on page 157
	db2_status	"Status of Database" on page 166
	idle_agents	"Number of Idle Agents" on page 208
	gw_total_cons	"Total Number of Attempted Connections for DB2 Connect" on page 432
	gw_cur_cons	"Current Number of Connections for DB2 Connect" on page 433
	gw_cons_wait_host	"Number of Connections Waiting for the Host to Reply" on page 433
	gw_cons_wait_client	"Number of Connections Waiting for the Client to Send Request" on page 434
	last_reset	"Last Reset Timestamp" on page 419
	max_agent_overflows	"Maximum Agent Overflows" on page 212
	num_gw_conn_switches	"Connection Switches" on page 213
	num_nodes_in_db2_instance	"Number of Nodes in Partition" on page 420
	post_threshold_hash_joins	"Hash Join Threshold" on page 227
	server_switch_list	Monitor switches control data
	smallest_log_avail_node	"Node with Least Available Log Space" on page 175
	sort_heap_top	"Sort Private Heap High Water Mark" on page 224

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
fcm	**buff_free**	"FCM Buffers Currently Free" on page 230
	buff_free_bottom	"Minimum FCM Buffers Free" on page 231
	ce_free	"Connection Entries Currently Free" on page 232
	ce_free_bottom	"Minimum Connection Entries" on page 233
	ma_free	"Message Anchors Currently Free" on page 231
	ma_free_bottom	"Minimum Message Anchors" on page 232
	node_number	"Node Number" on page 190
	rb_free	"Request Blocks Currently Free" on page 233
	rb_free_bottom	"Minimum Request Blocks" on page 234
fcm_node	**connection_status**	"Connection Status" on page 234
	node_number	"Node Number" on page 190
	total_buffers_sent	"Total FCM Buffers Sent" on page 235
	total_buffers_rcvd	"Total FCM Buffers Received" on page 235
memory_pool	**pool_cur_size**	"Current Size of Memory Pool" on page 215
	pool_id	"Memory Pool Identifier" on page 214
	pool_max_size	"Maximum Size of Memory Pool" on page 216
	node_number	"Node Number" on page 190
	pool_watermark	"Memory Pool Watermark" on page 217
dynsql_list	**db_name**	"Database Name" on page 162
	db_path	"Database Path" on page 163

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dynsql	**int_rows_deleted**	"Internal Rows Deleted" on page 357
	int_rows_inserted	"Internal Rows Inserted" on page 359
	int_rows_updated	"Internal Rows Updated" on page 358
	num_compilations	"Statement Compilations" on page 407
	num_executions	"Statement Executions" on page 406
	prep_time_best	"Statement Best Preparation Time" on page 408
	prep_time_worst	"Statement Worst Preparation Time" on page 407
	rows_read	"Rows Read" on page 355
	rows_written	"Rows Written" on page 354
	stmt_sorts	"Statement Sorts" on page 395
	stmt_text	"SQL Dynamic Statement Text" on page 394
	total_exec_time	"Elapsed Statement Execution Time" on page 408
	total_sys_cpu_time	"Total System CPU for a Statement" on page 417
	total_usr_cpu_time	"Total User CPU for a Statement" on page 418

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dbase	**active_sorts**	"Active Sorts" on page 224
	agents_top	"Number of Agents Created" on page 409
	appl_id_oldest_xact	"Application with Oldest Transaction" on page 174
	appls_cur_cons	"Applications Connected Currently" on page 204
	appls_in_db2	"Applications Executing in the Database Currently" on page 205
	binds_precompiles	"Binds/Precompiles Attempted" on page 384
	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 167
	catalog_node_name	"Catalog Node Network Name" on page 166
	commit_sql_stmts	"Commit Statements Attempted" on page 373
	connections_top	"Maximum Number of Concurrent Connections" on page 191
	coord_agents_top	"Maximum Number of Coordinating Agents" on page 209
	db_conn_time	"Database Activation Timestamp" on page 164
	db_heap_top	"Maximum Database Heap Allocated" on page 289
	db_location	"Database Location" on page 167
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	db_status	"Status of Database" on page 166
	ddl_sql_stmts	"Data Definition Language (DDL) SQL Statements" on page 378
	deadlocks	"Deadlocks Detected" on page 298

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dbase (continued)	**direct_read_reqs**	"Direct Read Requests" on page 269
	direct_read_time	"Direct Read Time" on page 270
	direct_reads	"Direct Reads From Database" on page 267
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_write_time	"Direct Write Time" on page 271
	direct_writes	"Direct Writes to Database" on page 268
	dynamic_sql_stmts	"Dynamic SQL Statements Attempted" on page 372
	failed_sql_stmts	"Failed Statement Operations" on page 372
	files_closed	"Database Files Closed" on page 248
	hash_join_overflows	"Hash Join Overflows" on page 229
	hash_join_small_overflows	"Hash Join Small Overflows" on page 229
	input_db_alias	"Input Database Alias" on page 419
	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 357
	int_rows_inserted	"Internal Rows Inserted" on page 359
	int_rows_updated	"Internal Rows Updated" on page 358
	last_backup	"Last Backup Timestamp" on page 168
	last_reset	"Last Reset Timestamp" on page 419
	lock_escals	"Number of Lock Escalations" on page 299
	lock_list_in_use	"Total Lock List Memory In Use" on page 298
	lock_timeouts	"Number of Lock Timeouts" on page 306

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dbase (continued)	**lock_wait_time**	"Time Waited On Locks" on page 317
	lock_waits	"Lock Waits" on page 316
	locks_held	"Locks Held" on page 297
	locks_waiting	"Current Agents Waiting On Locks" on page 318
	log_reads	"Number of Log Pages Read" on page 292
	log_writes	"Number of Log Pages Written" on page 293
	num_assoc_agents	"Number of Associated Agents" on page 212
	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 280
	pkg_cache_size_top	"Package Cache High Water Mark" on page 281
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 255
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 249
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 250
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 252
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 251
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 253
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 254
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 264
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_drty_pg_steal_clns	"Buffer Pool Victim Page Cleaners Triggered" on page 256
	pool_drty_pg_thrsh_clns	"Buffer Pool Threshold Cleaners Triggered" on page 257

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dbase (continued)	**pool_index_from_estore**	"Buffer Pool Index Pages from Extended Storage" on page 264
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_lsn_gap_clns	"Buffer Pool Log Space Cleaners Triggered" on page 256
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247
	prefetch_wait_time	"Time Waited for Prefetch" on page 259
	priv_workspace_num_overflows	"Private Workspace Overflows" on page 286
	priv_workspace_section_inserts	"Private Workspace Section Inserts" on page 288
	priv_workspace_section_lookups	"Private Workspace Section Lookups" on page 287
	priv_workspace_size_top	"Maximum Private Workspace Size" on page 285
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_deleted	"Rows Deleted" on page 351
	rows_inserted	"Rows Inserted" on page 352
	rows_read	"Rows Read" on page 355
	rows_selected	"Rows Selected" on page 353
	rows_updated	"Rows Updated" on page 352
	sec_log_used_top	"Maximum Secondary Log Space Used" on page 289
	sec_logs_allocated	"Secondary Logs Allocated Currently" on page 292
	select_sql_stmts	"Select SQL Statements Executed" on page 376
	server_platform	"Server Operating System" on page 160

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dbase (continued)	**shr_workspace_num_overflow**	"Shared Workspace Overflows" on page 283
	shr_workspace_section_inserts	"Shared Workspace Section Inserts" on page 285
	shr_workspace_section_lookups	"Shared Workspace Section Lookups" on page 284
	shr_workspace_size_top	"Maximum Shared Workspace Size" on page 282
	sort_heap_allocated	"Total Sort Heap Allocated" on page 218
	sort_overflows	"Sort Overflows" on page 223
	sort_shrheap_allocated	"Sort Share Heap Currently Allocated" on page 225
	sort_shrheap_top	"Sort Share Heap High Water Mark" on page 225
	static_sql_stmts	"Static SQL Statements Attempted" on page 371
	tot_log_used_top	"Maximum Total Log Space Used" on page 290
	total_cons	"Connects Since Database Activation" on page 203
	total_hash_joins	"Total Hash Joins" on page 227
	total_hash_loops	"Total Hash Loops" on page 228
	total_log_available	"Total Log Available" on page 295
	total_log_used	"Total Log Space Used" on page 294
	total_sec_cons	"Secondary Connections" on page 211
	total_sort_time	"Total Sort Time" on page 222
	total_sorts	"Total Sorts" on page 221
	uid_sql_stmts	"Update/Insert/Delete SQL Statements Executed" on page 377
	x_lock_escals	"Exclusive Lock Escalations" on page 301

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl_id_info	**agent_id**	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	appl_name	"Application Name" on page 175
	appl_status	"Application Status" on page 171
	auth_id	"Authorization ID" on page 179
	client_db_alias	"Database Alias Used by Application" on page 181
	client_nname	"Configuration NNAME of Client" on page 180
	client_prdid	"Client Product/Version ID" on page 180
	codepage_id	"ID of Code Page Used by Application" on page 173
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	input_db_alias	"Input Database Alias" on page 419
	sequence_no	"Sequence Number" on page 179
	status_change_time	"Application Status Change Time" on page 174
rollforward	**node_number**	"Node Number" on page 190
	rf_type	"Rollforward Type" on page 324
	rf_log_num	"Log Being Rolled Forward" on page 324
	rf_status	"Log Phase" on page 324
	rf_timestamp	"Rollforward Timestamp" on page 323
	ts_name	"Tablespace Being Rolled Forward" on page 323
table_list	**db_conn_time**	"Database Activation Timestamp" on page 164
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	input_db_alias	"Input Database Alias" on page 419
	last_reset	"Last Reset Timestamp" on page 419

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
table	**overflow_accesses**	"Accesses to Overflowed Records" on page 356
	page_reorgs	"Page Reorganizations" on page 360
	rows_read	"Rows Read" on page 355
	rows_written	"Rows Written" on page 354
	table_file_id	"Table File ID" on page 359
	table_name	"Table Name" on page 349
	table_schema	"Table Schema Name" on page 350
	table_type	"Table Type" on page 348
table_reorg	**reorg_completion**	"Table Reorganization Completion Flag" on page 364
	reorg_current_counter	"Table Reorganize Progress" on page 364
	reorg_end	"Table Reorganize End Time" on page 365
	reorg_index_id	"Index Used to Reorganize the Table" on page 365
	reorg_max_counter	"Total Amount of Table Reorganization" on page 364
	reorg_max_phase	"Maximum Table Reorganize Phase" on page 363
	reorg_phase	"Table Reorganize Phase" on page 362
	reorg_phase_start	"Table Reorganize Phase Start Time" on page 363
	reorg_start	"Table Reorganize Start Time" on page 365
	reorg_status	"Table Reorganize Status" on page 362
	reorg_tbspc_id	"Table Space Where Table is Reorganized" on page 366
	reorg_type	"Table Reorganize Attributes" on page 361
tablespace_list	**db_conn_time**	"Database Activation Timestamp" on page 164
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	input_db_alias	"Input Database Alias" on page 419
	last_reset	"Last Reset Timestamp" on page 419

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
tablespace	**direct_read_reqs**	"Direct Read Requests" on page 269
	direct_read_time	"Direct Read Time" on page 270
	direct_reads	"Direct Reads From Database" on page 267
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_write_time	"Direct Write Time" on page 271
	direct_writes	"Direct Writes to Database" on page 268
	files_closed	"Database Files Closed" on page 248
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 255
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 249
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 250
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 252
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 251
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 253
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 254
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 264
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 264
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
tablespace (continued)	**tablespace_content_type**	"Table Space Contents Type" on page 328
	tablespace_cur_pool_id	"Buffer Pool Currently Being Used" on page 330
	tablespace_extent_size	"Table Space Extent Size" on page 329
	tablespace_id	"Table Space Identification" on page 326
	tablespace_name	"Table Space Name" on page 326
	tablespace_next_pool_id	"Buffer Pool That Will Be Used at Next Startup" on page 330
	tablespace_page_size	"Table Space Page Size" on page 329
	tablespace_prefetch_size	"Table Space Prefetch Size" on page 330
	tablespace_type	"Table Space Type" on page 327

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
tablespace_nodeinfo	**tablespace_free_pages**	"Free Pages in Table Space" on page 332
	tablespace_min_recovery_time	"Minimum Recovery Time For Rollforward" on page 339
	tablespace_num_containers	"Number of Containers in Table Space" on page 340
	tablespace_num_quiescers	"Number of Quiescers" on page 336
	tablespace_num_ranges	"Number of Ranges in the Table Space Map" on page 343
	tablespace_page_top	"Table Space High Water Mark" on page 333
	tablespace_pending_free_pages	"Pending Free Pages in Table Space" on page 332
	tablespace_rebalancer_extents_processed	"Number of Extents the Rebalancer has Processed" on page 335
	tablespace_rebalancer_extents_remaining	"Total Number of Extents to be Processed by the Rebalancer" on page 334
	tablespace_rebalancer_last_extent_moved	"Last Extent Moved by the Rebalancer" on page 335
	tablespace_rebalancer_mode	"Rebalancer Mode" on page 333
	tablespace_rebalancer_priority	"Current Rebalancer Priority" on page 336
	tablespace_rebalancer_restart_time	"Rebalancer Restart Time" on page 334
	tablespace_rebalancer_start_time	"Rebalancer Start Time" on page 334
	tablespace_state	"Table Space State" on page 328
	tablespace_state_change_object_id	"State Change Object Identification" on page 338
	tablespace_state_change_ts_id	"State Change Table Space Identification" on page 339
	tablespace_total_pages	"Total Pages in Table Space" on page 331
	tablespace_usable_pages	"Usable Pages in Table Space" on page 331
	tablespace_used_pages	"Used Pages in Table Space" on page 332
tablespace_quiescer	**quiescer_agent_id**	"Quiescer Agent Identification" on page 337
	quiescer_auth_id	"Quiescer User Authorization Identification" on page 337
	quiescer_obj_id	"Quiescer Object Identification" on page 338
	quiescer_state	"Quiescer State" on page 338
	quiescer_ts_id	"Quiescer Table Space Identification" on page 337

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
tablespace_container	**container_accessible**	"Accessibility of Container" on page 343
	container_id	"Container Identification" on page 340
	container_name	"Container Name" on page 340
	container_stripe_set	"Stripe Set" on page 342
	container_total_pages	"Total Pages in Container" on page 341
	container_type	"Container Type" on page 341
	container_usable_pages	"Usable Pages in Container" on page 342
tablespace_range	**range_adjustment**	"Range Adjustment" on page 346
	range_container_id	"Range Container" on page 347
	range_end_stripe	"End Stripe" on page 346
	range_max_extent	"Maximum Extent in Range" on page 345
	range_max_page_number	"Maximum Page in Range" on page 345
	range_num_containers	"Number of Containers in Range" on page 346
	range_number	"Range Number" on page 344
	range_offset	"Range Offset" on page 347
	range_start_stripe	"Start Stripe" on page 345
	range_stripe_set_number	"Stripe Set Number" on page 344
db_lock_list	**appls_cur_cons**	"Applications Connected Currently" on page 204
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	input_db_alias	"Input Database Alias" on page 419
	locks_held	"Locks Held" on page 297
	locks_waiting	"Current Agents Waiting On Locks" on page 318

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl_lock_list	**agent_id**	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	appl_name	"Application Name" on page 175
	appl_status	"Application Status" on page 171
	auth_id	"Authorization ID" on page 179
	client_db_alias	"Database Alias Used by Application" on page 181
	codepage_id	"ID of Code Page Used by Application" on page 173
	locks_held	"Locks Held" on page 297
	locks_waiting	"Current Agents Waiting On Locks" on page 318
	lock_wait_time	"Time Waited On Locks" on page 317
	sequence_no	"Sequence Number" on page 179
	status_change_time	"Application Status Change Time" on page 174
lock_wait	**agent_id_holding_lk**	"Agent ID Holding Lock" on page 319
	appl_id_holding_lk	"Application ID Holding Lock" on page 320
	lock_attributes	"Lock Attributes" on page 312
	lock_current_mode	"Original Lock Mode Before Conversion" on page 315
	lock_escalation	"Lock Escalation" on page 308
	lock_mode	"Lock Mode" on page 302
	lock_mode_requested	"Lock Mode Requested" on page 309
	lock_name	"Lock Name" on page 312
	lock_object_type	"Lock Object Type Waited On" on page 304
	lock_release_flags	"Lock Release Flags" on page 313
	lock_wait_start_time	"Lock Wait Start Timestamp" on page 319
	node_number	"Node Number" on page 190
	subsection_number	"Subsection Number" on page 399
	table_name	"Table Name" on page 349
	table_schema	"Table Schema Name" on page 350
	tablespace_name	"Table Space Name" on page 326

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
lock	**lock_attributes**	"Lock Attributes" on page 312
	lock_count	"Lock Count" on page 313
	lock_current_mode	"Original Lock Mode Before Conversion" on page 315
	lock_escalation	"Lock Escalation" on page 308
	lock_hold_count	"Lock Hold Count" on page 314
	lock_mode	"Lock Mode" on page 302
	lock_name	"Lock Name" on page 312
	lock_object_name	"Lock Object Name" on page 305
	lock_object_type	"Lock Object Type Waited On" on page 304
	lock_release_flags	"Lock Release Flags" on page 313
	lock_status	"Lock Status" on page 303
	node_number	"Node Number" on page 190
	table_file_id	"Table File ID" on page 359
	table_name	"Table Name" on page 349
	table_schema	"Table Schema Name" on page 350
	tablespace_name	"Table Space Name" on page 326

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
bufferpool	**block_ios**	"Number of Block IO Requests" on page 260
	bp_cur_buffsz	"Current Size of Buffer Pool" on page 265
	bp_name	"Buffer Pool Name" on page 258
	bp_new_buffsz	"New Buffer Pool Size" on page 266
	bp_pages_left_to_remove	"Number of Pages Left to Remove" on page 266
	bp_tbsp_use_count	"Number of Table Spaces Mapped to Buffer Pool" on page 266
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	direct_read_reqs	"Direct Read Requests" on page 269
	direct_reads	"Direct Reads From Database" on page 267
	direct_read_time	"Direct Read Time" on page 270
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_writes	"Direct Writes to Database" on page 268
	direct_write_time	"Direct Write Time" on page 271
	files_closed	"Database Files Closed" on page 248
	input_db_alias	"Input Database Alias" on page 419
	pages_from_block_ios	"Total Number of Pages Read by Block IO" on page 260
	pages_from_vectored_ios	"Total Number of Pages Read by Vectored IO" on page 259
	physical_page_maps	"Number of Physical Page Maps" on page 260

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
bufferpool (continued)	**pool_async_data_reads**	"Buffer Pool Asynchronous Data Reads" on page 249
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 250
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 252
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 251
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 253
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 254
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 255
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 264
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 264
	vectored_ios	"Number of Vectored IO Requests" on page 259

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl_info	**agent_id**	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	appl_name	"Application Name" on page 175
	appl_status	"Application Status" on page 171
	auth_id	"Authorization ID" on page 179
	authority_lvl	"Authorization ID" on page 179
	client_db_alias	"Database Alias Used by Application" on page 181
	client_nname	"Configuration NNAME of Client" on page 180
	client_pid	"Client Process ID" on page 185
	client_platform	"Client Operating Platform" on page 185
	client_prdid	"Client Product/Version ID" on page 180
	client_protocol	"Client Communication Protocol" on page 186
	codepage_id	"ID of Code Page Used by Application" on page 173
	coord_agent_pid	"Coordinator Agent" on page 198
	coord_node_num	"Coordinating Node" on page 190
	corr_token	"DRDA Correlation Token" on page 184
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	execution_id	"User Login ID" on page 184
	input_db_alias	"Input Database Alias" on page 419
	num_assoc_agents	"Number of Associated Agents" on page 212
	sequence_no	"Sequence Number" on page 179
	status_change_time	"Application Status Change Time" on page 174
	territory_code	"Database Territory Code" on page 187
	tpmon_acc_str	"TP Monitor Client Accounting String" on page 466
	tpmon_client_app	"TP Monitor Client Application Name" on page 465
	tpmon_client_userid	"TP Monitor Client User ID" on page 464
	tpmon_client_wkstn	"TP Monitor Client Workstation Name" on page 465

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl	**acc_curs_blk**	"Accepted Block Cursor Requests" on page 368
	agent_sys_cpu_time	"System CPU Time used by Agent" on page 411
	agent_usr_cpu_time	"User CPU Time used by Agent" on page 411
	agents_stolen	"Stolen Agents" on page 210
	appl_con_time	"Connection Request Start Timestamp" on page 191
	appl_idle_time	"Application Idle Time" on page 197
	appl_priority	"Application Agent Priority" on page 187
	appl_priority_type	"Application Priority Type" on page 188
	associated_agents_top	"Maximum Number of Associated Agents" on page 211
	binds_precompiles	"Binds/Precompiles Attempted" on page 384
	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 373
	conn_complete_time	"Connection Request Completion Timestamp" on page 192
	ddl_sql_stmts	"Data Definition Language (DDL) SQL Statements" on page 378
	deadlocks	"Deadlocks Detected" on page 298
	direct_read_reqs	"Direct Read Requests" on page 269
	direct_read_time	"Direct Read Time" on page 270
	direct_reads	"Direct Reads From Database" on page 267
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_write_time	"Direct Write Time" on page 271
	direct_writes	"Direct Writes to Database" on page 268
	dynamic_sql_stmts	"Dynamic SQL Statements Attempted" on page 372
	failed_sql_stmts	"Failed Statement Operations" on page 372

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl (continued)	**hash_join_overflows**	"Hash Join Overflows" on page 229
	hash_join_small_overflows	"Hash Join Small Overflows" on page 229
	inbound_comm_address	"Inbound Communication Address" on page 439
	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 357
	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_escals	"Lock Escalation" on page 308
	lock_timeouts	"Number of Lock Timeouts" on page 306
	lock_wait_time	"Time Waited On Locks" on page 317
	lock_waits	"Lock Waits" on page 316
	locks_held	"Locks Held" on page 297
	locks_waiting	"Current Agents Waiting On Locks" on page 318
	num_agents	"Number of Agents Working on a Statement" on page 409
	open_loc_curs	"Open Local Cursors" on page 369
	open_loc_curs_blk	"Open Local Cursors with Blocking" on page 370
	open_rem_curs	"Open Remote Cursors" on page 366
	open_rem_curs_blk	"Open Remote Cursors with Blocking" on page 367
	pkg_cache_inserts	"Package Cache Inserts" on page 279
	pkg_cache_lookups	"Package Cache Lookups" on page 277
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 264

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl (continued)	**pool_data_l_reads**	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 264
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247
	prefetch_wait_time	"Time Waited for Prefetch" on page 259
	prev_uow_stop_time	"Previous Unit of Work Completion Timestamp" on page 192
	priv_workspace_num_overflows	"Private Workspace Overflows" on page 286
	priv_workspace_section_inserts	"Private Workspace Section Inserts" on page 288
	priv_workspace_section_lookups	"Private Workspace Section Lookups" on page 287
	priv_workspace_size_top	"Maximum Private Workspace Size" on page 285
	rej_curs_blk	"Rejected Block Cursor Requests" on page 368
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_deleted	"Rows Deleted" on page 351
	rows_inserted	"Rows Inserted" on page 352
	rows_read	"Rows Read" on page 355
	rows_selected	"Rows Selected" on page 353
	rows_updated	"Rows Updated" on page 352
	rows_written	"Rows Written" on page 354

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl (continued)	**select_sql_stmts**	"Select SQL Statements Executed" on page 376
	shr_workspace_num_overflow	"Shared Workspace Overflows" on page 283
	shr_workspace_section_inserts	"Shared Workspace Section Inserts" on page 285
	shr_workspace_section_lookups	"Shared Workspace Section Lookups" on page 284
	shr_workspace_size_top	"Maximum Shared Workspace Size" on page 282
	sort_overflows	"Sort Overflows" on page 223
	sql_reqs_since_commit	"SQL Requests Since Last Commit" on page 383
	static_sql_stmts	"Static SQL Statements Attempted" on page 371
	total_hash_joins	"Total Hash Joins" on page 227
	total_hash_loops	"Total Hash Loops" on page 228
	total_sort_time	"Total Sort Time" on page 222
	total_sorts	"Total Sorts" on page 221
	uid_sql_stmts	"Update/Insert/Delete SQL Statements Executed" on page 377
	uow_comp_status	"Unit of Work Completion Status" on page 195
	uow_elapsed_time	"Most Recent Unit of Work Elapsed Time" on page 195
	uow_lock_wait_time	"Total Time Unit of Work Waited on Locks" on page 318
	uow_log_space_used	"Unit of Work Log Space Used" on page 294
	uow_start_time	"Unit of Work Start Timestamp" on page 193
	uow_stop_time	"Unit of Work Stop Timestamp" on page 194
	x_lock_escals	"Exclusive Lock Escalations" on page 301

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
stmt	**agents_top**	"Number of Agents Created" on page 409
	blocking_cursor	"Blocking Cursor" on page 463
	consistency_token	"Package Consistency Token" on page 388
	creator	"Application Creator" on page 391
	cursor_name	"Cursor Name" on page 390
	degree_parallelism	"Degree of Parallelism" on page 410
	fetch_count	"Number of Successful Fetches" on page 396
	int_rows_deleted	"Internal Rows Deleted" on page 357
	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 409
	package_name	"Package Name" on page 387
	package_version_id	"Package Version" on page 388
	query_card_estimate	"Query Number of Rows Estimate" on page 397
	query_cost_estimate	"Query Cost Estimate" on page 398
	rows_read	"Rows Read" on page 355
	rows_written	"Rows Written" on page 354
	section_number	"Section Number" on page 389
	sort_overflows	"Sort Overflows" on page 223
	stmt_elapsed_time	"Most Recent Statement Elapsed Time" on page 393
	stmt_node_number	"Statement Node" on page 383
	stmt_operation	"Statement Operation" on page 386
	stmt_sorts	"Statement Sorts" on page 395
	stmt_start	"Statement Operation Start Timestamp" on page 391
	stmt_stop	"Statement Operation Stop Timestamp" on page 392
	stmt_sys_cpu_time	"System CPU Time used by Statement" on page 413
	stmt_text	"SQL Dynamic Statement Text" on page 394
	stmt_type	"Statement Type" on page 385
	stmt_usr_cpu_time	"User CPU Time used by Statement" on page 412
	total_sort_time	"Total Sort Time" on page 222

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
subsection	**rows_read**	"Rows Read" on page 355
	rows_written	"Rows Written" on page 354
	ss_exec_time	"Subsection Execution Elapsed Time" on page 401
	ss_node_number	"Subsection Node Number" on page 400
	ss_number	"Subsection Number" on page 399
	ss_status	"Subsection Status" on page 400
	ss_sys_cpu_time	"System CPU Time used by Subsection" on page 417
	ss_usr_cpu_time	"User CPU Time used by Subsection" on page 416
	tq_cur_send_spills	"Current Number of Tablequeue Buffers Overflowed" on page 403
	tq_id_waiting_on	"Waited on Node on a Tablequeue" on page 405
	tq_max_send_spills	"Maximum Number of Tablequeue Buffers Overflows" on page 405
	tq_node_waited_for	"Waited for Node on a Tablequeue" on page 402
	tq_rows_read	"Number of Rows Read from Tablequeues" on page 404
	tq_rows_written	"Number of Rows Written to Tablequeues" on page 404
	tq_tot_send_spills	"Total Number of Tablequeue Buffers Overflowed" on page 402
	tq_wait_for_any	"Waiting for Any Node to Send on a Tablequeue" on page 401
agent	**agent_pid**	"Process or Thread ID" on page 197

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_dbase	**commit_sql_stmts**	"Commit Statements Attempted" on page 373
	con_elapsed_time	"Most Recent Connection Elapsed Time" on page 461
	con_response_time	"Most Recent Response Time for Connect" on page 461
	dcs_db_name	"DCS Database Name" on page 430
	elapsed_exec_time	"Statement Execution Elapsed Time" on page 459
	failed_sql_stmts	"Failed Statement Operations" on page 372
	gw_comm_error_time	"Communication Error Time" on page 462
	gw_comm_errors	"Communication Errors" on page 461
	gw_con_time	"DB2 Connect Gateway First Connect Initiated" on page 432
	gw_connections_top	"Maximum Number of Concurrent Connections to Host Database" on page 432
	gw_cons_wait_client	"Number of Connections Waiting for the Client to Send Request" on page 434
	gw_cons_wait_host	"Number of Connections Waiting for the Host to Reply" on page 433
	gw_cur_cons	"Current Number of Connections for DB2 Connect" on page 433
	gw_total_cons	"Total Number of Attempted Connections for DB2 Connect" on page 432
	host_db_name	"Host Database Name" on page 431
	host_response_time	"Host Response Time" on page 459
	last_reset	"Last Reset Timestamp" on page 419

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_dbase (continued)	**max_data_sent_128**	"Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 444
	max_data_sent_256	"Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 445
	max_data_sent_512	"Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 446
	max_data_sent_1024	"Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 446
	max_data_sent_2048	"Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 447
	max_data_sent_4096	"Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 448
	max_data_sent_8192	"Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 449
	max_data_sent_16384	"Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 450
	max_data_sent_31999	"Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 451
	max_data_sent_64000	"Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 452
	max_data_sent_gt64000	"Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 453
	max_data_received_128	"Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 444
	max_data_received_256	"Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 445
	max_data_received_512	"Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 446
	max_data_received_1024	"Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 447
	max_data_received_2048	"Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 448
	max_data_received_4096	"Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 449
	max_data_received_8192	"Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 450

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_dbase (continued)	**max_data_received_16384**	"Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 450
	max_data_received_31999	"Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes" on page 451
	max_data_received_64000	"Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes" on page 452
	max_data_received_gt64000	"Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 453
	max_network_time_2_ms	"Number of Statements with Network Time of up to 2 ms" on page 454
	max_network_time_4_ms	"Number of Statements with Network Time between 2 and 4 ms" on page 454
	max_network_time_8_ms	"Number of Statements with Network Time between 4 and 8 ms" on page 455
	max_network_time_16_ms	"Number of Statements with Network Time between 8 and 16 ms" on page 455
	max_network_time_32_ms	"Number of Statements with Network Time between 16 and 32 ms" on page 456
	max_network_time_gt32_ms	"Number of Statements with Network Time greater than 32 ms" on page 456
	network_time_bottom	"Minimum Network Time for Statement" on page 458
	network_time_top	"Maximum Network Time for Statement" on page 457
	outbound_bytes_received	"Inbound Number of Bytes Received" on page 440
	outbound_bytes_sent	"Outbound Number of Bytes Sent" on page 440
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_selected	"Rows Selected" on page 353
	sql_stmts	"Number of SQL Statements Attempted" on page 435

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_appl_info	**agent_id**	"Application Handle (agent ID)" on page 169
	agent_status	"DCS Application Agents" on page 437
	appl_id	"Application ID" on page 176
	appl_name	"Application Name" on page 175
	auth_id	"Authorization ID" on page 179
	client_nname	"Configuration NNAME of Client" on page 180
	client_pid	"Client Process ID" on page 185
	client_platform	"Client Operating Platform" on page 185
	client_prdid	"Client Product/Version ID" on page 180
	client_protocol	"Client Communication Protocol" on page 186
	codepage_id	"ID of Code Page Used by Application" on page 173
	dcs_appl_status	"DCS Application Status" on page 436
	dcs_db_name	"DCS Database Name" on page 430
	execution_id	"User Login ID" on page 184
	gw_db_alias	"Database Alias at the Gateway" on page 431
	host_ccsid	"Host Coded Character Set ID" on page 438
	host_db_name	"Host Database Name" on page 431
	host_prdid	"Host Product/Version ID" on page 182
	inbound_comm_address	"Inbound Communication Address" on page 439
	outbound_appl_id	"Outbound Application ID" on page 182
	outbound_comm_address	"Outbound Communication Address" on page 439
	outbound_comm_protocol	"Outbound Communication Protocol" on page 438
	outbound_sequence_no	"Outbound Sequence Number" on page 183
	sequence_no	"Sequence Number" on page 179
	status_change_time	"Application Status Change Time" on page 174

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_appl	**appl_idle_time**	"Application Idle Time" on page 197
	commit_sql_stmts	"Commit Statements Attempted" on page 373
	elapsed_exec_time	"Statement Execution Elapsed Time" on page 459
	failed_sql_stmts	"Failed Statement Operations" on page 372
	gw_con_time	"DB2 Connect Gateway First Connect Initiated" on page 432
	gw_exec_time	"Elapsed Time Spent on DB2 Connect Gateway Processing" on page 434
	host_response_time	"Host Response Time" on page 459
	inbound_bytes_received	"Inbound Number of Bytes Received" on page 440
	inbound_bytes_sent	"Inbound Number of Bytes Sent" on page 442
	last_reset	"Last Reset Timestamp" on page 419
	open_cursors	"Number of Open Cursors" on page 436
	outbound_bytes_received	"Outbound Number of Bytes Received" on page 441
	outbound_bytes_sent	"Outbound Number of Bytes Sent" on page 440
	prev_uow_stop_time	"Previous Unit of Work Completion Timestamp" on page 192
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_selected	"Rows Selected" on page 353
	sql_stmts	"Number of SQL Statements Attempted" on page 435
	tpmon_acc_str	"TP Monitor Client Accounting String" on page 466
	tpmon_client_app	"TP Monitor Client Application Name" on page 465
	tpmon_client_userid	"TP Monitor Client User ID" on page 464
	tpmon_client_wkstn	"TP Monitor Client Workstation Name" on page 465
	uow_comp_status	"Unit of Work Completion Status" on page 195
	uow_elapsed_time	"Most Recent Unit of Work Elapsed Time" on page 195
	uow_start_time	"Unit of Work Start Timestamp" on page 193
	uow_stop_time	"Unit of Work Stop Timestamp" on page 194
	xid	"Transaction ID" on page 458

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_appl (continued)	**max_data_sent_128**	"Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 444
	max_data_sent_256	"Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 445
	max_data_sent_512	"Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 446
	max_data_sent_1024	"Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 446
	max_data_sent_2048	"Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 447
	max_data_sent_4096	"Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 448
	max_data_sent_8192	"Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 449
	max_data_sent_16384	"Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 450
	max_data_sent_31999	"Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 451
	max_data_sent_64000	"Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 452
	max_data_sent_gt64000	"Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 453
	max_data_received_128	"Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 444
	max_data_received_256	"Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 445
	max_data_received_512	"Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 446
	max_data_received_1024	"Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 447
	max_data_received_2048	"Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 448
	max_data_received_4096	"Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 449
	max_data_received_8192	"Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 450

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_appl (continued)	**max_data_received_16384**	"Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 450
	max_data_received_31999	"Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes" on page 451
	max_data_received_64000	"Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes" on page 452
	max_data_received_gt64000	"Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 453
	max_network_time_2_ms	"Number of Statements with Network Time of up to 2 ms" on page 454
	max_network_time_4_ms	"Number of Statements with Network Time between 2 and 4 ms" on page 454
	max_network_time_8_ms	"Number of Statements with Network Time between 4 and 8 ms" on page 455
	max_network_time_16_ms	"Number of Statements with Network Time between 8 and 16 ms" on page 455
	max_network_time_32_ms	"Number of Statements with Network Time between 16 and 32 ms" on page 456
	max_network_time_gt32_ms	"Number of Statements with Network Time greater than 32 ms" on page 456
	network_time_top	"Maximum Network Time for Statement" on page 457
	network_time_bottom	"Minimum Network Time for Statement" on page 458

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dcs_stmt	**blocking_cursor**	"Blocking Cursor" on page 463
	creator	"Application Creator" on page 391
	elapsed_exec_time	"Statement Execution Elapsed Time" on page 459
	fetch_count	"Number of Successful Fetches" on page 396
	gw_exec_time	"Elapsed Time Spent on DB2 Connect Gateway Processing" on page 434
	host_response_time	"Host Response Time" on page 459
	inbound_bytes_received	"Inbound Number of Bytes Received" on page 440
	inbound_bytes_sent	"Inbound Number of Bytes Sent" on page 442
	num_transmissions	"Number of Transmissions" on page 460
	outbound_blocking_cursor	"Outbound Blocking Cursor" on page 463
	outbound_bytes_received	"Outbound Number of Bytes Received" on page 441
	outbound_bytes_sent	"Outbound Number of Bytes Sent" on page 440
	package_name	"Package Name" on page 387
	query_card_estimate	"Query Number of Rows Estimate" on page 397
	query_cost_estimate	"Query Cost Estimate" on page 398
	section_number	"Section Number" on page 389
	stmt_elpased_time	"Most Recent Statement Elapsed Time" on page 393
	stmt_operation	"Statement Operation" on page 386
	stmt_start	"Statement Operation Start Timestamp" on page 391
	stmt_stop	"Statement Operation Stop Timestamp" on page 392
	stmt_text	"SQL Dynamic Statement Text" on page 394

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
stmt_transmissions	**elapsed_exec_time**	"Statement Execution Elapsed Time" on page 459
	host_response_time	"Host Response Time" on page 459
	outbound_bytes_received	"Outbound Number of Bytes Received" on page 441
	outbound_bytes_sent	"Outbound Number of Bytes Sent" on page 440
	outbound_bytes_sent_top	"Maximum Outbound Number of Bytes Sent" on page 442
	outbound_bytes_received_top	"Maximum Outbound Number of Bytes Received" on page 443
	outbound_bytes_sent_bottom	"Minimum Outbound Number of Bytes Sent" on page 443
	outbound_bytes_received_bottom	"Minimum Outbound Number of Bytes Received" on page 443
	sql_stmts	"Number of SQL Statements Attempted" on page 435

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
stmt_transmissions (continued)	**max_data_sent_128**	"Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes" on page 444
	max_data_sent_256	"Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes" on page 445
	max_data_sent_512	"Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes" on page 446
	max_data_sent_1024	"Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes" on page 446
	max_data_sent_2048	"Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes" on page 447
	max_data_sent_4096	"Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes" on page 448
	max_data_sent_8192	"Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes" on page 449
	max_data_sent_16384	"Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes" on page 450
	max_data_sent_31999	"Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes" on page 451
	max_data_sent_64000	"Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes" on page 452
	max_data_sent_gt64000	"Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes" on page 453
	max_data_received_128	"Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes" on page 444
	max_data_received_256	"Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes" on page 445
	max_data_received_512	"Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes" on page 446
	max_data_received_1024	"Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes" on page 447
	max_data_received_2048	"Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes" on page 448
	max_data_received_4096	"Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes" on page 449
	max_data_received_8192	"Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes" on page 450

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
stmt_transmissions (continued)	**max_data_received_16384**	"Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes" on page 450
	max_data_received_31999	"Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes" on page 451
	max_data_received_64000	"Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes" on page 452
	max_data_received_gt64000	"Number of Statements with Outbound Bytes Received Greater than 64000 Bytes" on page 453
	max_network_time_2_ms	"Number of Statements with Network Time of up to 2 ms" on page 454
	max_network_time_4_ms	"Number of Statements with Network Time between 2 and 4 ms" on page 454
	max_network_time_8_ms	"Number of Statements with Network Time between 4 and 8 ms" on page 455
	max_network_time_16_ms	"Number of Statements with Network Time between 8 and 16 ms" on page 455
	max_network_time_32_ms	"Number of Statements with Network Time between 16 and 32 ms" on page 456
	max_network_time_gt32_ms	"Number of Statements with Network Time greater than 32 ms" on page 456
	network_time_top	"Maximum Network Time for Statement" on page 457
	network_time_bottom	"Minimum Network Time for Statement" on page 458

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
dbase_remote	**commit_sql_stmts**	"Commit Statements Attempted" on page 373
	create_nickname	"Create Nicknames" on page 470
	create_nickname_time	"Create Nickname Response Time" on page 475
	datasource_name	"Data Source Name" on page 467
	db_name	"Database Name" on page 162
	delete_sql_stmts	"Deletes" on page 469
	delete_time	"Delete Response Time" on page 474
	disconnects	"Disconnects" on page 467
	failed_sql_stmts	"Failed Statement Operations" on page 372
	insert_sql_stmts	"Inserts" on page 468
	insert_time	"Insert Response Time" on page 473
	passthru_time	"Pass-Through Time" on page 476
	passthrus	"Pass-Through" on page 470
	remote_lock_time	"Remote Lock Time" on page 477
	remote_locks	"Remote Locks" on page 471
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_deleted	"Rows Deleted" on page 351
	rows_inserted	"Rows Inserted" on page 352
	rows_selected	"Rows Selected" on page 353
	rows_updated	"Rows Updated" on page 352
	select_sql_stmts	"Select SQL Statements Executed" on page 376
	select_time	"Query Response Time" on page 472
	sp_rows_selected	"Rows Returned by Stored Procedures" on page 472
	stored_proc_time	"Stored Procedure Time" on page 476
	stored_procs	"Stored Procedures" on page 471
	total_cons	"Connects Since Database Activation" on page 203
	update_sql_stmts	"Updates" on page 468
	update_time	"Update Response Time" on page 474

Table 11. Snapshot Monitor Logical Data Groups and Monitor Elements (continued)

Snapshot logical data groups	Monitor element name	Monitor element title
appl_remote	**commit_sql_stmts**	"Commit Statements Attempted" on page 373
	create_nickname	"Create Nicknames" on page 470
	create_nickname_time	"Create Nickname Response Time" on page 475
	datasource_name	"Data Source Name" on page 467
	db_name	"Database Name" on page 162
	delete_sql_stmts	"Deletes" on page 469
	delete_time	"Delete Response Time" on page 474
	failed_sql_stmts	"Failed Statement Operations" on page 372
	insert_sql_stmts	"Inserts" on page 468
	insert_time	"Insert Response Time" on page 473
	passthru_time	"Pass-Through Time" on page 476
	passthrus	"Pass-Through" on page 470
	remote_lock_time	"Remote Lock Time" on page 477
	remote_locks	"Remote Locks" on page 471
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_deleted	"Rows Deleted" on page 351
	rows_inserted	"Rows Inserted" on page 352
	rows_selected	"Rows Selected" on page 353
	rows_updated	"Rows Updated" on page 352
	select_sql_stmts	"Select SQL Statements Executed" on page 376
	select_time	"Query Response Time" on page 472
	sp_rows_selected	"Rows Returned by Stored Procedures" on page 472
	stored_proc_time	"Stored Procedure Time" on page 476
	stored_procs	"Stored Procedures" on page 471
	update_sql_stmts	"Updates" on page 468
	update_time	"Update Response Time" on page 474

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 12. Event Monitor Data Stream: Monitor Section

Event type	Logical data group	Available information
Monitor Level	event_log_stream_header	Identifies the version level and byte order of the event monitor. Applications can use this header to determine whether they can handle the evmon output stream.

Prolog:

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

Table 13. 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 14. 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.

Table 14. Event Monitor Data Stream: Contents Section (continued)

Event type	Logical data group	Available information
Overflow	event_overflow	Number of records lost - generated when writer cannot keep up with a (non-blocked) event monitor.

Epilog:

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

Table 15. 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:

- "Event monitors" on page 45
- "Database system monitor data organization" on page 4

Related tasks:

- "Collecting information about database system events" on page 47

Related reference:

- "Event monitor sample output" on page 71

Event monitor logical data groups and monitor elements

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

Table 16. Event Monitor Logical Data Groups and Monitor Elements

Event logical data groups	Monitor element name	Monitor element title
event_db	**binds_precompiles**	"Binds/Precompiles Attempted" on page 384
	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 167
	catalog_node_name	"Catalog Node Network Name" on page 166
	commit_sql_stmts	"Commit Statements Attempted" on page 373
	connections_top	"Maximum Number of Concurrent Connections" on page 191
	db_heap_top	"Maximum Database Heap Allocated" on page 289
	ddl_sql_stmts	"Data Definition Language (DDL) SQL Statements" on page 378
	deadlocks	"Deadlocks Detected" on page 298
	direct_read_reqs	"Direct Read Requests" on page 269
	direct_read_time	"Direct Read Time" on page 270
	direct_reads	"Direct Reads From Database" on page 267
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_write_time	"Direct Write Time" on page 271
	direct_writes	"Direct Writes to Database" on page 268
	disconn_time	"Database Deactivation Timestamp" on page 165
	dynamic_sql_stmts	"Dynamic SQL Statements Attempted" on page 372
	evmon_activates	"Number of Event Monitor Activations" on page 426
	evmon_flushes	"Number of Event Monitor Flushes" on page 425
	failed_sql_stmts	"Failed Statement Operations" on page 372
	files_closed	"Database Files Closed" on page 248

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_db (continued)	**hash_join_overflows**	"Hash Join Overflows" on page 229
	hash_join_small_overflows	"Hash Join Small Overflows" on page 229
	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 357
	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 299
	lock_timeouts	"Number of Lock Timeouts" on page 306
	lock_wait_time	"Time Waited On Locks" on page 317
	lock_waits	"Lock Waits" on page 316
	log_reads	"Number of Log Pages Read" on page 292
	log_writes	"Number of Log Pages Written" on page 293
	partial_record	"Partial Record" on page 424
	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 280
	pkg_cache_size_top	"Package Cache High Water Mark" on page 281
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 255
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 249
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 250
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 252
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 251
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 253
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 254

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_db (continued)	**pool_data_from_estore**	"Buffer Pool Data Pages from Extended Storage" on page 264
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_drty_pg_steal_clns	"Buffer Pool Victim Page Cleaners Triggered" on page 256
	pool_drty_pg_thrsh_clns	"Buffer Pool Threshold Cleaners Triggered" on page 257
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 264
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_lsn_gap_clns	"Buffer Pool Log Space Cleaners Triggered" on page 256
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247
	prefetch_wait_time	"Time Waited for Prefetch" on page 259
	priv_workspace_num_overflows	"Private Workspace Overflows" on page 286
	priv_workspace_section_inserts	"Private Workspace Section Inserts" on page 288
	priv_workspace_section_lookups	"Private Workspace Section Lookups" on page 287
	priv_workspace_size_top	"Maximum Private Workspace Size" on page 285
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_deleted	"Rows Deleted" on page 351
	rows_inserted	"Rows Inserted" on page 352
	rows_read	"Rows Read" on page 355
	rows_selected	"Rows Selected" on page 353
	rows_updated	"Rows Updated" on page 352

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_db (continued)	**sec_log_used_top**	"Maximum Secondary Log Space Used" on page 289
	select_sql_stmts	"Select SQL Statements Executed" on page 376
	server_platform	"Server Operating System" on page 160
	shr_workspace_num_overflow	"Shared Workspace Overflows" on page 283
	shr_workspace_section_inserts	"Shared Workspace Section Inserts" on page 285
	shr_workspace_section_lookups	"Shared Workspace Section Lookups" on page 284
	shr_workspace_size_top	"Maximum Shared Workspace Size" on page 282
	sort_overflows	"Sort Overflows" on page 223
	static_sql_stmts	"Static SQL Statements Attempted" on page 371
	tot_log_used_top	"Maximum Total Log Space Used" on page 290
	total_cons	"Connects Since Database Activation" on page 203
	total_hash_joins	"Total Hash Joins" on page 227
	total_hash_loops	"Total Hash Loops" on page 228
	total_sort_time	"Total Sort Time" on page 222
	total_sorts	"Total Sorts" on page 221
	uid_sql_stmts	"Update/Insert/Delete SQL Statements Executed" on page 377
	x_lock_escals	"Exclusive Lock Escalations" on page 301
event_dbheader	**conn_time**	"Time of Database Connection" on page 165
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_connheader	**agent_id**	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	appl_name	"Application Name" on page 175
	auth_id	"Authorization ID" on page 179
	client_db_alias	"Database Alias Used by Application" on page 181
	client_nname	"Configuration NNAME of Client" on page 180
	client_pid	"Client Process ID" on page 185
	client_platform	"Client Operating Platform" on page 185
	client_prdid	"Client Product/Version ID" on page 180
	client_protocol	"Client Communication Protocol" on page 186
	codepage_id	"ID of Code Page Used by Application" on page 173
	conn_time	"Time of Database Connection" on page 165
	corr_token	"DRDA Correlation Token" on page 184
	execution_id	"User Login ID" on page 184
	node_number	"Node Number" on page 190
	sequence_no	"Sequence Number" on page 179
	territory_code	"Database Territory Code" on page 187
event_start	**start_time**	"Event Start Time" on page 393
event_deadlock	**deadlock_id**	"Deadlock Event Identifier" on page 309
	deadlock_node	"Partition Number Where Deadlock Occurred" on page 310
	dl_conns	"Connections Involved in Deadlock" on page 308
	evmon_activates	"Number of Event Monitor Activations" on page 426
	rolled_back_agent_id	"Rolled Back Agent" on page 322
	rolled_back_appl_id	"Rolled Back Application" on page 321
	rolled_back_participant_no	"Rolled Back Application Participant" on page 311
	rolled_back_sequence_no	"Rolled Back Sequence Number" on page 322
	start_time	"Event Start Time" on page 393

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_dlconn	**agent_id**	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	appl_id_holding_lk	"Application ID Holding Lock" on page 320
	deadlock_id	"Deadlock Event Identifier" on page 309
	deadlock_node	"Partition Number Where Deadlock Occurred" on page 310
	evmon_activates	"Number of Event Monitor Activations" on page 426
	lock_attributes	"Lock Attributes" on page 312
	lock_count	"Lock Count" on page 313
	lock_current_mode	"Original Lock Mode Before Conversion" on page 315
	lock_escalation	"Lock Escalation" on page 308
	lock_hold_count	"Lock Hold Count" on page 314
	lock_mode	"Lock Mode" on page 302
	lock_mode_requested	"Lock Mode Requested" on page 309
	lock_name	"Lock Name" on page 312
	lock_node	"Lock Node" on page 306
	lock_object_name	"Lock Object Name" on page 305
	lock_object_type	"Lock Object Type Waited On" on page 304
	lock_release_flags	"Lock Release Flags" on page 313
	lock_wait_start_time	"Lock Wait Start Timestamp" on page 319
	participant_no	"Participant within Deadlock" on page 310
	participant_no_holding_lk	"Participant Holding a Lock on the Object Required by Application" on page 310
	sequence_no	"Sequence Number" on page 179
	sequence_no_holding_lk	"Sequence Number Holding Lock" on page 321
	start_time	"Event Start Time" on page 393
	table_name	"Table Name" on page 349
	table_schema	"Table Schema Name" on page 350
	tablespace_name	"Table Space Name" on page 326

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_detailed_dlconn	**agent_id**	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	appl_id_holding_lk	"Application ID Holding Lock" on page 320
	blocking_cursor	"Blocking Cursor" on page 463
	consistency_token	"Package Consistency Token" on page 388
	creator	"Application Creator" on page 391
	cursor_name	"Cursor Name" on page 390
	deadlock_id	"Deadlock Event Identifier" on page 309
	deadlock_node	"Partition Number Where Deadlock Occurred" on page 310
	evmon_activates	"Number of Event Monitor Activations" on page 426
	lock_escalation	"Lock Escalation" on page 308
	lock_mode	"Lock Mode" on page 302
	lock_mode_requested	"Lock Mode Requested" on page 309
	lock_node	"Lock Node" on page 306
	lock_object_name	"Lock Object Name" on page 305
	lock_object_type	"Lock Object Type Waited On" on page 304
	lock_wait_start_time	"Lock Wait Start Timestamp" on page 319
	locks_held	"Locks Held" on page 297
	locks_in_list	"Number of Locks Reported" on page 311
	package_name	"Package Name" on page 387
	package_version_id	"Package Version" on page 388
	participant_no	"Participant within Deadlock" on page 310
	participant_no_holding_lk	"Participant Holding a Lock on the Object Required by Application" on page 310
	section_number	"Section Number" on page 389
	sequence_no	"Sequence Number" on page 179
	sequence_no_holding_lk	"Sequence Number Holding Lock" on page 321
	start_time	"Event Start Time" on page 393

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_detailed_dlconn (continued)	**stmt_operation**	"Statement Operation" on page 386
	stmt_text	"SQL Dynamic Statement Text" on page 394
	stmt_type	"Statement Type" on page 385
	table_name	"Table Name" on page 349
	table_schema	"Table Schema Name" on page 350
	tablespace_name	"Table Space Name" on page 326
lock	**lock_attributes**	"Lock Attributes" on page 312
	lock_count	"Lock Count" on page 313
	lock_current_mode	"Original Lock Mode Before Conversion" on page 315
	lock_escalation	"Lock Escalation" on page 308
	lock_hold_count	"Lock Hold Count" on page 314
	lock_mode	"Lock Mode" on page 302
	lock_name	"Lock Name" on page 312
	lock_object_name	"Lock Object Name" on page 305
	lock_object_type	"Lock Object Type Waited On" on page 304
	lock_release_flags	"Lock Release Flags" on page 313
	lock_status	"Lock Status" on page 303
	node_number	"Node Number" on page 190
	table_file_id	"Table File ID" on page 359
	table_name	"Table Name" on page 349
	table_schema	"Table Schema Name" on page 350
	tablespace_name	"Table Space Name" on page 326

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_table	**event_time**	"Event Time" on page 425
	evmon_activates	"Number of Event Monitor Activations" on page 426
	evmon_flushes	"Number of Event Monitor Flushes" on page 425
	overflow_accesses	"Accesses to Overflowed Records" on page 356
	page_reorgs	"Page Reorganizations" on page 360
	partial_record	"Partial Record" on page 424
	rows_read	"Rows Read" on page 355
	rows_written	"Rows Written" on page 354
	table_name	"Table Name" on page 349
	table_schema	"Table Schema Name" on page 350
	table_type	"Table Type" on page 348
event_tablespace	**direct_read_reqs**	"Direct Read Requests" on page 269
	direct_read_time	"Direct Read Time" on page 270
	direct_reads	"Direct Reads From Database" on page 267
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_write_time	"Direct Write Time" on page 271
	direct_writes	"Direct Writes to Database" on page 268

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_tablespace (continued)	**event_time**	"Event Time" on page 425
	evmon_activates	"Number of Event Monitor Activations" on page 426
	evmon_flushes	"Number of Event Monitor Flushes" on page 425
	files_closed	"Database Files Closed" on page 248
	partial_record	"Partial Record" on page 424
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 255
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 249
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 250
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 252
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 251
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 253
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 254
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 264
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 264
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247
	tablespace_name	"Table Space Name" on page 326

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_conn	**acc_curs_blk**	"Accepted Block Cursor Requests" on page 368
	agent_id	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	appl_priority	"Application Agent Priority" on page 187
	appl_priority_type	"Application Priority Type" on page 188
	authority_lvl	"User Authorization Level" on page 189
	binds_precompiles	"Binds/Precompiles Attempted" on page 384
	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 373
	coord_node	"Secondary Connections" on page 211
	ddl_sql_stmts	"Data Definition Language (DDL) SQL Statements" on page 378
	deadlocks	"Deadlocks Detected" on page 298
	direct_read_reqs	"Direct Read Requests" on page 269
	direct_read_time	"Direct Read Time" on page 270
	direct_reads	"Direct Reads From Database" on page 267
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_write_time	"Direct Write Time" on page 271
	direct_writes	"Direct Writes to Database" on page 268
	disconn_time	"Database Deactivation Timestamp" on page 165
	dynamic_sql_stmts	"Dynamic SQL Statements Attempted" on page 372
	failed_sql_stmts	"Failed Statement Operations" on page 372
	hash_join_overflows	"Hash Join Overflows" on page 229
	hash_join_small_overflows	"Hash Join Small Overflows" on page 229
	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

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_conn (continued)	**int_rows_deleted**	"Internal Rows Deleted" on page 357
	int_rows_inserted	"Internal Rows Inserted" on page 359
	int_rows_updated	"Internal Rows Updated" on page 358
	lock_escals	"Lock Escalation" on page 308
	lock_timeouts	"Number of Lock Timeouts" on page 306
	lock_wait_time	"Time Waited On Locks" on page 317
	lock_waits	"Lock Waits" on page 316
	partial_record	"Partial Record" on page 424
	pkg_cache_inserts	"Package Cache Inserts" on page 279
	pkg_cache_lookups	"Package Cache Lookups" on page 277
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 264
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 264
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_conn (continued)	**prefetch_wait_time**	"Time Waited for Prefetch" on page 259
	priv_workspace_num_overflows	"Private Workspace Overflows" on page 286
	priv_workspace_section_inserts	"Private Workspace Section Inserts" on page 288
	priv_workspace_section_lookups	"Private Workspace Section Lookups" on page 287
	priv_workspace_size_top	"Maximum Private Workspace Size" on page 285
	rej_curs_blk	"Rejected Block Cursor Requests" on page 368
	rollback_sql_stmts	"Rollback Statements Attempted" on page 375
	rows_deleted	"Internal Rows Deleted" on page 357
	rows_inserted	"Internal Rows Inserted" on page 359
	rows_read	"Rows Read" on page 355
	rows_selected	"Rows Selected" on page 353
	rows_updated	"Internal Rows Updated" on page 358
	rows_written	"Rows Written" on page 354
	select_sql_stmts	"Select SQL Statements Executed" on page 376
	sequence_no	"Sequence Number" on page 179
	shr_workspace_num_overflow	"Shared Workspace Overflows" on page 283
	shr_workspace_section_inserts	"Shared Workspace Section Inserts" on page 285
	shr_workspace_section_lookups	"Shared Workspace Section Lookups" on page 284
	shr_workspace_size_top	"Maximum Shared Workspace Size" on page 282
	sort_overflows	"Sort Overflows" on page 223
	static_sql_stmts	"Static SQL Statements Attempted" on page 371
	system_cpu_time	"System CPU Time" on page 415
	total_hash_joins	"Total Hash Joins" on page 227
	total_hash_loops	"Total Hash Loops" on page 228
	total_sort_time	"Total Sort Time" on page 222
	total_sorts	"Total Sorts" on page 221
	uid_sql_stmts	"Update/Insert/Delete SQL Statements Executed" on page 377
	user_cpu_time	"User CPU Time" on page 414
	x_lock_escals	"Exclusive Lock Escalations" on page 301

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
sqlca	**sqlcabc**	sqlcabc
	sqlcode	sqlcode
	sqlerrml	sqlerrml
	sqlcaid	sqlcaid
	sqlerrmc	sqlerrmc
	sqlerrp	sqlerrp
	sqlerrd	sqlerrd
	sqlwarn	sqlwarn
	sqlstate	sqlstate

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_stmt	**agent_id**	"Application Handle (agent ID)" on page 169
	agents_top	"Number of Agents Created" on page 409
	appl_id	"Application ID" on page 176
	blocking_cursor	"Blocking Cursor" on page 463
	consistency_token	"Package Consistency Token" on page 388
	creator	"Application Creator" on page 391
	cursor_name	"Cursor Name" on page 390
	fetch_count	"Number of Successful Fetches" on page 396
	int_rows_deleted	"Internal Rows Deleted" on page 357
	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 424
	rows_read	"Rows Read" on page 355
	rows_written	"Rows Written" on page 354
	section_number	"Section Number" on page 389
	sequence_no	"Sequence Number" on page 179
	sort_overflows	"Sort Overflows" on page 223
	sql_req_id	"Request Identifier for SQL Statement" on page 426
	sqlca	"SQL Communications Area (SQLCA)" on page 397
	start_time	"Event Start Time" on page 393
	stmt_operation	"Statement Operation" on page 386
	stmt_text	"SQL Dynamic Statement Text" on page 394
	stmt_type	"Statement Type" on page 385
	stop_time	"Event Stop Time" on page 392
	system_cpu_time	"System CPU Time" on page 415
	total_sort_time	"Total Sort Time" on page 222
	total_sorts	"Total Sorts" on page 221
	user_cpu_time	"User CPU Time" on page 414

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_subsection	**agent_id**	"Application Handle (agent ID)" on page 169
	num_agents	"Number of Agents Working on a Statement" on page 409
	partial_record	"Partial Record" on page 424
	ss_exec_time	"Subsection Execution Elapsed Time" on page 401
	ss_node_number	"Subsection Node Number" on page 400
	ss_number	"Subsection Number" on page 399
	ss_sys_cpu_time	"System CPU Time used by Subsection" on page 417
	ss_usr_cpu_time	"User CPU Time used by Subsection" on page 416
	tq_max_send_spills	"Maximum Number of Tablequeue Buffers Overflows" on page 405
	tq_rows_read	"Number of Rows Read from Tablequeues" on page 404
	tq_rows_written	"Number of Rows Written to Tablequeues" on page 404
	tq_tot_send_spills	"Total Number of Tablequeue Buffers Overflowed" on page 402

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_xact	**agent_id**	"Application Handle (agent ID)" on page 169
	appl_id	"Application ID" on page 176
	lock_escals	"Number of Lock Escalations" on page 299
	lock_wait_time	"Time Waited On Locks" on page 317
	locks_held_top	"Maximum Number of Locks Held" on page 307
	partial_record	"Partial Record" on page 424
	prev_uow_stop_time	"Previous Unit of Work Completion Timestamp" on page 192
	rows_read	"Rows Read" on page 355
	rows_written	"Rows Written" on page 354
	sequence_no	"Sequence Number" on page 179
	system_cpu_time	"System CPU Time" on page 415
	uow_log_space_used	"Unit of Work Log Space Used" on page 294
	uow_start_time	"Unit of Work Start Timestamp" on page 193
	uow_status	"Unit of Work Status" on page 196
	uow_stop_time	"Unit of Work Stop Timestamp" on page 194
	user_cpu_time	"User CPU Time" on page 414
	x_lock_escal	"Exclusive Lock Escalations" on page 301
event_bufferpool	**bp_name**	"Buffer Pool Name" on page 258
	db_name	"Database Name" on page 162
	db_path	"Database Path" on page 163
	direct_read_reqs	"Direct Read Requests" on page 269
	direct_read_time	"Direct Read Time" on page 270
	direct_reads	"Direct Reads From Database" on page 267
	direct_write_reqs	"Direct Write Requests" on page 270
	direct_write_time	"Direct Write Time" on page 271
	direct_writes	"Direct Writes to Database" on page 268

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_bufferpool (continued)	**event_time**	"Event Time" on page 425
	evmon_activates	"Number of Event Monitor Activations" on page 426
	evmon_flushes	"Number of Event Monitor Flushes" on page 425
	files_closed	"Database Files Closed" on page 248
	partial_record	"Partial Record" on page 424
	pool_async_data_read_reqs	"Buffer Pool Asynchronous Read Requests" on page 255
	pool_async_data_reads	"Buffer Pool Asynchronous Data Reads" on page 249
	pool_async_data_writes	"Buffer Pool Asynchronous Data Writes" on page 250
	pool_async_index_reads	"Buffer Pool Asynchronous Index Reads" on page 252
	pool_async_index_writes	"Buffer Pool Asynchronous Index Writes" on page 251
	pool_async_read_time	"Buffer Pool Asynchronous Read Time" on page 253
	pool_async_write_time	"Buffer Pool Asynchronous Write Time" on page 254
	pool_data_from_estore	"Buffer Pool Data Pages from Extended Storage" on page 264
	pool_data_l_reads	"Buffer Pool Data Logical Reads" on page 239
	pool_data_p_reads	"Buffer Pool Data Physical Reads" on page 240
	pool_data_to_estore	"Buffer Pool Data Pages to Extended Storage" on page 262
	pool_data_writes	"Buffer Pool Data Writes" on page 241
	pool_index_from_estore	"Buffer Pool Index Pages from Extended Storage" on page 264
	pool_index_l_reads	"Buffer Pool Index Logical Reads" on page 243
	pool_index_p_reads	"Buffer Pool Index Physical Reads" on page 244
	pool_index_to_estore	"Buffer Pool Index Pages to Extended Storage" on page 263
	pool_index_writes	"Buffer Pool Index Writes" on page 245
	pool_read_time	"Total Buffer Pool Physical Read Time" on page 246
	pool_write_time	"Total Buffer Pool Physical Write Time" on page 247
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 190

Table 16. Event Monitor Logical Data Groups and Monitor Elements (continued)

Event logical data groups	Monitor element name	Monitor element title
event_log_header	**byte_order**	"Byte Order of Event Data" on page 422
	codepage_id	"ID of Code Page Used by Application" on page 173
	event_monitor_name	"Event Monitor Name" on page 424
	num_nodes_in_db2_instance	"Number of Nodes in Partition" on page 420
	server_prdid	"Server Product/Version ID" on page 159
	server_instance_name	"Server Instance Name" on page 157
	territory_code	"Database Territory Code" on page 187
	version	"Version of Monitor Data" on page 423

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 uppercased 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. level.

- *Monitor switch*: The system monitor switch that must be set to obtain this information. If the switch is Basic, data will always be collected for the monitor element.

Event monitoring information

If a monitor element is collected by event monitors, a table with the following fields is shown.

- *Event type*: The level of information that can be collected by the event monitor. The event monitor must be created with this event type to collect this information. For example, the appl_status monitor element is collected for CONNECTIONS event monitors.
- *Logical data grouping*: The logical data group where captured event information is returned. If parsing the data stream directly, the logical data group identifier is uppercased 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:

- Start Database Manager Timestamp
- Configuration NNAME at Monitoring (Server) Node
- Server Instance Name
- Database Manager Type at Monitored (Server) Node
- Server Product/Version ID
- Server Version
- Service Level
- Server Operating System
- Product Name
- Status of DB2 Instance

- Time Zone Displacement

Start Database Manager Timestamp

Element identifier db2start_time

Element type timestamp

Table 17. 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:

- "Snapshot Time" on page 420

Configuration NNAME at Monitoring (Server) Node

Element identifier server_nname

Element type information

Table 18. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Description

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

Usage This element can be used to identify the database server node 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 nodes. This node name is determined based on the *nname* configuration parameter.

Related reference:

- "Configuration NNAME of Client" on page 180

Server Instance Name

Element identifier server_instance_name

Element type information

Table 19. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	collected	Basic

Table 20. 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_nname*, this information can be useful if you are saving your monitor output in a file or database for later analysis, and you need to differentiate the data from different instances of the database manager.

Related reference:

• "Configuration NNAME at Monitoring (Server) Node" on page 157

Database Manager Type at Monitored (Server) Node

Element identifier server_db2_type

Element type information

Table 21. 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	Command Line Processor Output
sqlf_nt_server	Database server with local and remote clients
sqlf_nt_stand_req	Database server with local clients

The API symbolic constants are defined in the include file *sqlutil.h*.

Related reference:

- "Configuration NNAME at Monitoring (Server) Node" on page 157

Server Product/Version ID

Element identifier server_prdid

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 product and version that is running on the server.

Usage It is in the form PPPVVRRM, where:

PPP is SQL

VV identifies a 2-digit version number (with high-order 0 in the case of a 1-digit version)

RR identifies a 2-digit release number (with high-order 0 in the case of a 1-digit release)

M identifies a 1-digit modification level

Related reference:

- "Client Product/Version ID" on page 180

Server Version

Element identifier server_version

Element type information

Table 24. 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 Product/Version ID" on page 159

Service Level

Element identifier service_level

Element type information

Table 25. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description

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

Server Operating System

Element identifier server_platform

Element type information

Table 26. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 27. 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:

- "Database Location" on page 167
- "Client Operating Platform" on page 185

Product Name

Element identifier product_name

Element type information

Table 28. 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.

Status of DB2 Instance

Element identifier db2_status

Element type information

Table 29. 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.

The value returned is always SQLM_DB2_ACTIVE.

Related reference:

- "Status of Database" on page 166

Time Zone Displacement

Element identifier time_zone_disp

Element type information

Table 30. 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:

- Database Name
- Database Path
- Database Activation Timestamp
- Time of Database Connection
- Database Deactivation Timestamp
- Status of Database
- Catalog Node Network Name
- Database Location
- Catalog Node Number
- Last Backup Timestamp

Database Name

Element identifier db_name

Element type information

Table 31. Snapshot Monitoring Information

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

Table 31. Snapshot Monitoring Information (continued)

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

Table 32. 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:

- "Database Path" on page 163
- "Database Alias Used by Application" on page 181
- "Last Reset Timestamp" on page 419
- "Input Database Alias" on page 419

Database Path

Element identifier db_path

Element type information

Table 33. 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 34. 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:

- "Database Name" on page 162
- "Input Database Alias" on page 419

Database Activation Timestamp

Element identifier db_conn_time

Element type timestamp

Table 35. 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:

- "Connection Request Start Timestamp" on page 191
- "Snapshot Time" on page 420
- "Time of Database Connection" on page 165

Time of Database Connection

Element identifier conn_time

Element type timestamp

Table 36. 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:

- "Database Activation Timestamp" on page 164
- "Database Deactivation Timestamp" on page 165

Database Deactivation Timestamp

Element identifier disconn_time

Element type timestamp

Table 37. 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).

Status of Database

Element identifier db_status

Element type information

Table 38. 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	Description
SQLM_DB_ACTIVE	The database is active.
SQLM_DB_QUIESCE_PEND	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	The database has been quiesced. New connections to the database are **not** permitted and new units of work **cannot** be started.
SQLM_DB_ROLLFWD	A rollforward is in progress on the database.

Related reference:
- "Status of DB2 Instance" on page 161

Catalog Node Network Name

Element identifier catalog_node_name

Element type information

Table 39. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 40. 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.

Database Location

Element identifier db_location

Element type information

Table 41. 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 Operating System" on page 160

Catalog Node Number

Element identifier catalog_node

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 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.

Last Backup Timestamp

Element identifier last_backup

Element type timestamp

Table 44. 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.

Application identification and status

Application identification and status monitor elements

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

- Application Handle (agent ID)
- Application Status
- ID of Code Page Used by Application
- Application Status Change Time
- Application with Oldest Transaction
- Node with Least Available Log Space
- Application Name
- Application ID
- Sequence Number
- Authorization ID

- Configuration NNAME of Client
- Client Product/Version ID
- Database Alias Used by Application
- Host Product/Version ID
- Outbound Application ID
- Outbound Sequence Number
- User Login ID
- DRDA Correlation Token
- Client Process ID
- Client Operating Platform
- Client Communication Protocol
- Database Territory Code
- Application Agent Priority
- Application Priority Type
- User Authorization Level
- Node Number
- Coordinating Node
- Connection Request Start Timestamp
- Maximum Number of Concurrent Connections
- Connection Request Completion Timestamp
- Previous Unit of Work Completion Timestamp
- Unit of Work Start Timestamp
- Unit of Work Stop Timestamp
- Most Recent Unit of Work Elapsed Time
- Unit of Work Completion Status
- Unit of Work Status
- Previous Transaction Stop Time
- Application Idle Time
- DB2 agent information data elements

Application Handle (agent ID)

Element identifier agent_id

Element type information

Table 45. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_id_info	Basic

Table 45. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	appl_lock_list	Basic
DCS Application	dcs_appl_info	Basic

Table 46. Event Monitoring Information

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

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 Universal Database (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 Universal 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.

Application Status

Element identifier appl_status

Element type information

Table 47. Snapshot Monitoring Information

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

Table 48. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connection	event_conn	-

Description

The current status of the application.

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.

API Constant	Description
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_TPREP	**Transaction Prepared:** The unit of work is part of a global transaction that has entered the prepared phase of the two-phase commit protocol.
SQLM_THCOMT	**Transaction Heuristically Committed:** The unit of work is part of a global transaction that has been heuristically committed.
SQLM_THABRT	**Transaction Heuristically Rolled Back:** The unit of work is part of a global transaction that has been heuristically rolled-back.
SQLM_TEND	**Transaction Ended:** The unit of work is part of a global transaction that has ended but has not yet entered the prepared phase of the two-phase commit protocol.
SQLM_CREATE_DB	**Creating Database:** The agent has initiated a request to create a database and that request has not yet completed.
SQLM_RESTART	**Restarting Database:** The application is restarting a database in order to perform crash recovery.
SQLM_RESTORE	**Restoring Database:** The application is restoring a backup image to the database.
SQLM_BACKUP	**Backing Up Database:** The application is performing a backup of the database.
SQLM_LOAD	**Data Fast Load:** The application is performing a “fast load” of data into the database.

API Constant	Description
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.

Related reference:

- "Application Status Change Time" on page 174
- "Statement Operation" on page 386

ID of Code Page Used by Application

Element identifier codepage_id

Element type information

Table 49. 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 50. 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.

Application Status Change Time

Element identifier status_change_time

Element type timestamp

Table 51. 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:

- "Application Status" on page 171

Application with Oldest Transaction

Element identifier appl_id_oldest_xact

Element type information

Table 52. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

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:

- "Deadlocks Detected" on page 298
- "Agent ID Holding Lock" on page 319

Node with Least Available Log Space

Element identifier smallest_log_avail_node

Element type information

Table 53. 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:

- "Total Log Space Used" on page 294
- "Total Log Available" on page 295
- "Application with Oldest Transaction" on page 174

Application Name

Element identifier appl_name

Element type information

Table 54. 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 55. 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 manager or DB2 Connect.

Usage This element may 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 to establish the database connection. For DRDA-AS connections, this name is the DRDA external name.

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:

- "ID of Code Page Used by Application" on page 173
- "Application ID" on page 176

Application ID

Element identifier appl_id

Element type information

Table 56. 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 57. 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.

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 and/or DDCS 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`

   Example `A12CF9E8.3F0A.930131214645`

Details A TCP/IP-generated application ID is made up by concatenating the IP address in hexadecimal characters, the port number (4 hexadecimal characters), and a unique identifier for the instance of this application. The IP address is a 32-bit number displayed as a maximum of 8 hexadecimal characters.

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 `MMDDHHMMSS`.

4. NetBIOS

   Format `*NETBIOS.nname.Application instance`

   Example `*NETBIOS.SBOIVIN.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.

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 Application ID" on page 182
- "Client Communication Protocol" on page 186

Sequence Number

Element identifier sequence_no

Element type information

Table 58. Snapshot Monitoring Information

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

Table 59. 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 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.

Authorization ID

Element identifier auth_id

Element type information

Table 60. Snapshot Monitoring Information

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

Table 61. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
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:

- "Application Name" on page 175

Configuration NNAME of Client

Element identifier client_nname

Element type information

Table 62. Snapshot Monitoring Information

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

Table 63. 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 node.

Usage You can use this element to identify the client node that is running the application.

Related reference:

- "Configuration NNAME at Monitoring (Server) Node" on page 157

Client Product/Version ID

Element identifier client_prdid

Element type information

Table 64. Snapshot Monitoring Information

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

Table 65. 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 PPPVVRRM, 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 Product/Version ID" on page 159

Database Alias Used by Application

Element identifier client_db_alias

Element type information

Table 66. Snapshot Monitoring Information

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

Table 67. 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 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:

- "Database Name" on page 162
- "Last Reset Timestamp" on page 419
- "Input Database Alias" on page 419

Host Product/Version ID

Element identifier host_prdid

Element type information

Table 68. 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 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 Application ID

Element identifier outbound_appl_id

Element type information

Table 69. 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.

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.

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:

- "Application ID" on page 176

Outbound Sequence Number

Element identifier outbound_sequence_no

Element type information

Table 70. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl_info	Basic

Description

This element is reserved for future use. In this release, its value will always be "0001". It may contain different values in future releases of the product.

User Login ID

Element identifier execution_id

Element type information

Table 71. Snapshot Monitoring Information

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

Table 72. 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:

- "Authorization ID" on page 179

DRDA Correlation Token

Element identifier corr_token

Element type information

Table 73. Snapshot Monitoring Information

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

Table 74. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-

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 Process ID

Element identifier client_pid

Element type information

Table 75. Snapshot Monitoring Information

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

Table 76. 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 Operating Platform

Element identifier client_platform

Element type information

Table 77. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_info	Basic

Table 77. Snapshot Monitoring Information (continued)

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

Table 78. 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 Operating System" on page 160

Client Communication Protocol

Element identifier client_protocol

Element type information

Table 79. Snapshot Monitoring Information

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

Table 80. 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.

Database Territory Code

Element identifier territory_code

Element type information

Table 81. Snapshot Monitoring Information

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

Table 82. 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.

Application Agent Priority

Element identifier appl_priority

Element type information

Table 83. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 84. 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:

• "Application Priority Type" on page 188

Application Priority Type

Element identifier appl_priority_type

Element type information

Table 85. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 86. 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:

- "Application Agent Priority" on page 187
- "Query Cost Estimate" on page 398

User Authorization Level

Element identifier authority_lvl

Element type information

Table 87. Snapshot Monitoring Information

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

Table 88. 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_NOT_FENC
- SQL_SYSCTRL
- SQL_SYSMAINT

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

- SQL_SYSADM_GRP
- SQL_DBADM_GRP
- SQL_CREATETAB_GRP
- SQL_BINDADD_GRP
- SQL_CONNECT_GRP

- SQL_CREATE_NOT_FENC_GRP
- SQL_SYSCTRL_GRP
- SQL_SYSMAINT_GRP

Node Number

Element identifier node_number

Element type information

Table 89. 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
Table Space	rollforward	Basic
Lock	lock	Basic
Lock	lock_wait	Basic

Table 90. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Connections	event_connheader	-
Deadlocks	lock	-
Overflow Record	event_overflow	-

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.

Coordinating Node

Element identifier coord_node

Element type information

Table 91. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 92. 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.

Connection Request Start Timestamp

Element identifier appl_con_time

Element type timestamp

Table 93. 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:

- "Connection Request Completion Timestamp" on page 192

Maximum Number of Concurrent Connections

Element identifier connections_top

Element type water mark

Table 94. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 95. 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:

```
rem_cons_in + local_cons
```

Related reference:

- "Remote Connections To Database Manager" on page 200
- "Local Connections" on page 202

Connection Request Completion Timestamp

Element identifier conn_complete_time

Element type timestamp

Table 96. 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:

- "Connection Request Start Timestamp" on page 191

Previous Unit of Work Completion Timestamp

Element identifier prev_uow_stop_time

Element type timestamp

Table 97. 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:

- "Connection Request Completion Timestamp" on page 192
- "Unit of Work Start Timestamp" on page 193
- "Unit of Work Stop Timestamp" on page 194

Unit of Work Start Timestamp

Element identifier uow_start_time

Element type timestamp

Table 98. 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 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:

- "Connection Request Completion Timestamp" on page 192
- "Previous Unit of Work Completion Timestamp" on page 192
- "Unit of Work Stop Timestamp" on page 194

Unit of Work Stop Timestamp

Element identifier uow_stop_time

Element type timestamp

Table 99. 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 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:

- "Connection Request Completion Timestamp" on page 192
- "Previous Unit of Work Completion Timestamp" on page 192
- "Unit of Work Start Timestamp" on page 193

Most Recent Unit of Work Elapsed Time

Element identifier uow_elapsed_time

Element type time

Table 100. 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:

- "Communication Errors" on page 461
- "Communication Error Time" on page 462

Unit of Work Completion Status

Element identifier uow_comp_status

Element type information

Table 101. Snapshot Monitoring Information

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

Table 102. 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.

Unit of Work Status

Element identifier uow_status

Element type information

Table 103. 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.

Previous Transaction Stop Time

Event Type	Logical Data Grouping
Transaction	event_xact
Element Name	prev_stop_time
Element Type	timestamp

Description: The completion time of the last unit of work.

Usage: You may use this element to calculate the time spent in the application between units of work.

This is the unit of work that completed prior to the unit of work for which this transaction event is generated.

For applications within their first unit of work, this is the database connection request completion time.

Application Idle Time

Element identifier appl_idle_time

Element type information

Table 104. Snapshot Monitoring Information

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

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:

- "Application Agent Priority" on page 187
- "Application Priority Type" on page 188
- "Query Cost Estimate" on page 398

DB2 agent information

DB2 agent information monitor elements

The following database system monitor elements provide information about agents:

- Process or Thread ID
- Coordinator Agent

Process or Thread ID

Element identifier agent_pid

Element type information

Table 105. 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:

- "Coordinator Agent" on page 198

Coordinator Agent

Element identifier coord_agent_pid

Element type information

Table 106. 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:

- "Process or Thread ID" on page 197

Database manager configuration

Database manager configuration monitor elements

The following elements provide database manager configuration information.

- Agents and connections data elements
- Memory pool data elements
- Sort data elements
- Hash join data elements
- Fast communications manager data 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.

The following elements provide agent and connection information:
- Remote Connections To Database Manager
- Remote Connections Executing in the Database Manager
- Local Connections
- Local Connections Executing in the Database Manager
- Local Databases with Current Connects
- Connects Since Database Activation
- Applications Connected Currently
- Applications Executing in the Database Currently
- Agents Registered
- Agents Waiting for a Token
- Maximum Number of Agents Registered
- Maximum Number of Agents Waiting
- Number of Idle Agents
- Agents Assigned From Pool
- Agents Created Due to Empty Agent Pool
- Maximum Number of Coordinating Agents
- Stolen Agents
- Maximum Number of Associated Agents
- Committed Private Memory
- Secondary Connections
- Number of Associated Agents
- Maximum Agent Overflows
- Connection Switches

Remote Connections To Database Manager

Element identifier rem_cons_in

Element type gauge

Table 107. 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:

- "Remote Connections Executing in the Database Manager" on page 201
- "Local Connections" on page 202
- "Local Connections Executing in the Database Manager" on page 202

Remote Connections Executing in the Database Manager

Element identifier rem_cons_in_exec

Element type gauge

Table 108. 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:

- "Remote Connections To Database Manager" on page 200
- "Local Connections" on page 202
- "Local Connections Executing in the Database Manager" on page 202

Local Connections

Element identifier local_cons

Element type gauge

Table 109. 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.

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:

- "Remote Connections To Database Manager" on page 200
- "Remote Connections Executing in the Database Manager" on page 201
- "Local Connections Executing in the Database Manager" on page 202

Local Connections Executing in the Database Manager

Element identifier local_cons_in_exec

Element type gauge

Table 110. 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 *maxcagents* configuration parameter, described in the *Administration Guide*.

Related reference:

- "Remote Connections To Database Manager" on page 200
- "Remote Connections Executing in the Database Manager" on page 201
- "Local Connections" on page 202

Local Databases with Current Connects

Element identifier con_local_dbases

Element type gauge

Table 111. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

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

Connects Since Database Activation

Element identifier total_cons

Element type counter

Table 112. 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 113. 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:

- "Database Activation Timestamp" on page 164
- "Applications Connected Currently" on page 204
- "Applications Executing in the Database Currently" on page 205
- "Secondary Connections" on page 211

Applications Connected Currently

Element identifier appls_cur_cons

Element type gauge

Table 114. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
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:

- "Remote Connections To Database Manager" on page 200
- "Local Connections" on page 202
- "Connects Since Database Activation" on page 203
- "Applications Executing in the Database Currently" on page 205

Applications Executing in the Database Currently

Element identifier appls_in_db2

Element type gauge

Table 115. 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:

- "Remote Connections Executing in the Database Manager" on page 201
- "Local Connections Executing in the Database Manager" on page 202
- "Connects Since Database Activation" on page 203
- "Applications Connected Currently" on page 204
- "Current Agents Waiting On Locks" on page 318

Agents Registered

Element identifier agents_registered

Element type gauge

Table 116. 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:

• "Maximum Number of Agents Registered" on page 206

Agents Waiting for a Token

Element identifier agents_waiting_on_token

Element type gauge

Table 117. 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" on page 205

Maximum Number of Agents Registered

Element identifier agents_registered_top

Element type water mark

Table 118. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

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" on page 205
- "Maximum Number of Agents Waiting" on page 207

Maximum Number of Agents Waiting

Element identifier agents_waiting_top

Element type water mark

Table 119. 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 for a Token" on page 206

- "Maximum Number of Agents Registered" on page 206

Number of Idle Agents

Element identifier idle_agents

Element type gauge

Table 120. 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.

Related reference:

- "Agents Registered" on page 205
- "Maximum Number of Agents Registered" on page 206
- "Maximum Number of Agents Waiting" on page 207

Agents Assigned From Pool

Element identifier agents_from_pool

Element type counter

Table 121. 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

```
Agents Created Due to Empty Agent Pool / 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 Due to Empty Agent Pool" on page 209
- "Maximum Number of Coordinating Agents" on page 209

Agents Created Due to Empty Agent Pool

Element identifier agents_created_empty_pool

Element type counter

Table 122. 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)*.

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 Assigned From Pool" on page 208
- "Maximum Number of Coordinating Agents" on page 209

Maximum Number of Coordinating Agents

Element identifier coord_agents_top

Element type water mark

Table 123. 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 Assigned From Pool" on page 208
- "Agents Created Due to Empty Agent Pool" on page 209

Stolen Agents

Element identifier agents_stolen

Element type counter

Table 124. 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.

Related reference:

- "Number of Agents Working on a Statement" on page 409

Maximum Number of Associated Agents

Element identifier associated_agents_top

Element type water mark

Table 125. 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 Assigned From Pool" on page 208
- "Agents Created Due to Empty Agent Pool" on page 209

Committed Private Memory

Element identifier comm_private_mem

Element type gauge

Table 126. 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).

Secondary Connections

Element identifier total_sec_cons

Element type counter

Table 127. 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:

- "Start Database Manager Timestamp" on page 157
- "Database Activation Timestamp" on page 164
- "Connects Since Database Activation" on page 203

Number of Associated Agents

Element identifier num_assoc_agents

Element type gauge

Table 128. 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 Assigned From Pool" on page 208
- "Agents Created Due to Empty Agent Pool" on page 209
- "Maximum Number of Associated Agents" on page 211
- "Maximum Agent Overflows" on page 212

Maximum Agent Overflows

Element identifier max_agent_overflows

Element type gauge

Table 129. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

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 Assigned From Pool" on page 208
- "Agents Created Due to Empty Agent Pool" on page 209
- "Maximum Number of Associated Agents" on page 211
- "Number of Associated Agents" on page 212

Connection Switches

Element identifier num_gw_conn_switches

Element type gauge

Table 130. 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 in conjunction with inactive_gw_agents to determine if the size of the agent pool should be increased.

Related reference:

- "Agents Registered" on page 205
- "Maximum Number of Agents Registered" on page 206
- "Secondary Connections" on page 211

- "Number of Associated Agents" on page 212
- "Maximum Agent Overflows" on page 212

Memory pool

Memory pool monitor elements

The following elements provide information about the memory pools:

- Memory Pool Identifier
- Current Size of Memory Pool
- Maximum Size of Memory Pool
- Memory Pool Watermark

The nature of memory_pool data elements varies between platforms. The difference being that 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. 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.

Memory Pool Identifier

Element identifier pool_id

Element type Information

Table 131. Snapshot Monitoring Information

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

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

API Constant	Description
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

Related reference:

- "Current Size of Memory Pool" on page 215
- "Maximum Size of Memory Pool" on page 216
- "Memory Pool Watermark" on page 217

Current Size of Memory Pool

Element identifier pool_cur_size

Element type Information

Table 132. Snapshot Monitoring Information

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

Description

The current size of a memory pool.

Usage To track system memory usage, use this value in conjunction with pool_max_size, pool_id, and pool_watermark.

To see if a memory pool is nearly full, compare pool_max_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_max_size, you may want to consider increasing the size of the utility heap.

Related reference:

- "Memory Pool Identifier" on page 214
- "Maximum Size of Memory Pool" on page 216
- "Memory Pool Watermark" on page 217

Maximum Size of Memory Pool

Element identifier pool_max_size

Element type Information

Table 133. Snapshot Monitoring Information

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

Description

The maximum size of a memory pool.

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_max_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_max_size, you may want to consider increasing the size of the utility heap.

Related reference:

- "Memory Pool Identifier" on page 214
- "Current Size of Memory Pool" on page 215
- "Memory Pool Watermark" on page 217

Memory Pool Watermark

Element identifier pool_watermark

Element type Information

Table 134. Snapshot Monitoring Information

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

Description

The largest size of a memory pool since its creation.

Usage On continuously running systems you can use the pool_watermark and pool_max_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_max_size values. If you observe that the value of pool_watermark is becoming increasingly close to pool_max_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:

- "Memory Pool Identifier" on page 214
- "Current Size of Memory Pool" on page 215
- "Maximum Size of Memory Pool" on page 216

Sort

Sort monitor elements

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

- Total Sort Heap Allocated
- Post Threshold Sorts
- Piped Sorts Requested
- Piped Sorts Accepted
- Total Sorts
- Total Sort Time
- Sort Overflows
- Active Sorts
- Sort Share Heap Currently Allocated
- Sort Share Heap High Water Mark

Total Sort Heap Allocated

Element identifier sort_heap_allocated

Element type gauge

Table 135. 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 database configuration parameter *sortheap*.

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.

Related reference:

- "Total Sorts" on page 221

Post Threshold Sorts

Element identifier post_threshold_sorts

Element type counter

Table 136. 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 (*sheapthres* configuration parameter), the database manager will allocate sort heap using a value less than that specified by the *sortheap* configuration parameter.

Each active sort on the system allocates memory, which may result in sorting taking up too much of the system memory available. Sorts that start after the sort heap threshold has been reached may not receive an optimum amount of memory to execute, but, as a result, the entire system may benefit. By modifying the sort heap threshold and sort heap size configuration parameters, the performance of sort operations and/or the overall system can be improved. If this element's value is high, you can:

- Increase the sort heap threshold (*sheapthres*) or,
- Adjust applications to use fewer or smaller sorts via SQL query changes.

Related reference:

- "Active Sorts" on page 224
- "Statement Sorts" on page 395

Piped Sorts Requested

Element identifier piped_sorts_requested

Element type counter

Table 137. 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.

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

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

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

Related reference:

- "Post Threshold Sorts" on page 219
- "Piped Sorts Accepted" on page 220

Piped Sorts Accepted

Element identifier piped_sorts_accepted

Element type counter

Table 138. 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:

- "Post Threshold Sorts" on page 219
- "Piped Sorts Requested" on page 219

Total Sorts

Element identifier total_sorts

Element type counter

Table 139. 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 140. 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:

- "Total Sort Time" on page 222
- "Sort Overflows" on page 223

Total Sort Time

Element identifier total_sort_time

Element type counter

Table 141. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Sort
Application	appl	Sort
Application	stmt	Sort

For snapshot monitoring, this counter can be reset.

Table 142. 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" on page 221
- "Sort Overflows" on page 223

Sort Overflows

Element identifier sort_overflows

Element type counter

Table 143. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Application	appl	Basic
Application	stmt	Basic

For snapshot monitoring, this counter can be reset.

Table 144. 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" on page 221

Active Sorts

Element identifier active_sorts

Element type gauge

Table 145. 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:

• "Total Sort Heap Allocated" on page 218
• "Total Sorts" on page 221

Sort Private Heap High Water Mark

Element identifier sort_heap_top

Element type water mark

Table 146. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Description
The private sort memory high-water mark 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 Share Heap Currently Allocated

Element identifier sort_shrheap_allocated

Element type information

Table 147. 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 Share Heap High Water Mark

Element identifier sort_shrheap_top

Element type water mark

Table 148. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Description

Database-wide shared sort memory high-water mark.

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 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.

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
- Hash Join Threshold
- Total Hash Loops
- Hash Join Overflows
- Hash Join Small Overflows

Total Hash Joins

Element identifier total_hash_joins

Element type counter

Table 149. 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 150. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
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:

- "Hash Join Threshold" on page 227
- "Total Hash Loops" on page 228
- "Hash Join Overflows" on page 229
- "Hash Join Small Overflows" on page 229

Hash Join Threshold

Element identifier post_threshold_hash_joins

Element type counter

Table 151. 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" on page 227
- "Total Hash Loops" on page 228
- "Hash Join Overflows" on page 229
- "Hash Join Small Overflows" on page 229

Total Hash Loops

Element identifier total_hash_loops

Element type counter

Table 152. 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 153. 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" on page 227
- "Hash Join Threshold" on page 227
- "Hash Join Overflows" on page 229
- "Hash Join Small Overflows" on page 229

Hash Join Overflows

Element identifier hash_join_overflows

Element type counter

Table 154. 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 155. 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" on page 227
- "Hash Join Threshold" on page 227
- "Total Hash Loops" on page 228
- "Hash Join Small Overflows" on page 229

Hash Join Small Overflows

Element identifier hash_join_small_overflows

Element type counter

Table 156. 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 157. 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 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" on page 227
- "Hash Join Threshold" on page 227
- "Total Hash Loops" on page 228
- "Hash Join Overflows" on page 229

Fast communications manager

Fast communications manager monitor elements

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

- FCM Buffers Currently Free
- Minimum FCM Buffers Free
- Message Anchors Currently Free
- Minimum Message Anchors
- Connection Entries Currently Free
- Minimum Connection Entries
- Request Blocks Currently Free
- Minimum Request Blocks
- Number of Nodes
- Connection Status
- Total FCM Buffers Sent
- Total FCM Buffers Received

FCM Buffers Currently Free

Element identifier buff_free

Element type gauge

Table 158. 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:

- "Minimum FCM Buffers Free" on page 231

Minimum FCM Buffers Free

Element identifier buff_free_bottom

Element type water mark

Table 159. 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:

- "FCM Buffers Currently Free" on page 230

Message Anchors Currently Free

Element identifier ma_free

Element type gauge

Table 160. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description
This element indicates the number of message anchors currently free.

Usage Use the number of message anchors currently free in conjunction with the *fcm_num_anchors* configuration parameter to determine the current message anchor utilization. You can use this information to tune *fcm_num_anchors*.

Related reference:

- "Minimum Message Anchors" on page 232

Minimum Message Anchors

Element identifier ma_free_bottom

Element type water mark

Table 161. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description
The lowest number of free message anchors reached during processing.

Usage Use this element in conjunction with the *fcm_num_anchors* configuration parameter to determine the maximum message anchors utilization. If ma_free_bottom is low, you should increase *fcm_num_anchors* to ensure that operations do not run out of message anchors. If ma_free_bottom is high, you can decrease *fcm_num_anchors* to conserve system resources.

Related reference:

- "Message Anchors Currently Free" on page 231

Connection Entries Currently Free

Element identifier ce_free

Element type gauge

Table 162. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description

This element indicates the number of connection entries currently free.

Usage Use the number of connection entries currently free in conjunction with the *fcm_num_connect* configuration parameter to determine the current connection entry utilization. You can use this information to tune *fcm_num_connect*.

Related reference:

- "Minimum Connection Entries" on page 233

Minimum Connection Entries

Element identifier ce_free_bottom

Element type water mark

Table 163. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description

The lowest number of free connection entries reached during processing.

Usage Use this element in conjunction with the *fcm_num_connect* configuration parameter to determine the maximum connection entry utilization. If ce_free_bottom is low, you should increase *fcm_num_connect* to ensure that operations do not run out of connection entries. If ce_free_bottom is high, you can decrease *fcm_num_connect* to conserve system resources.

Related reference:

- "Connection Entries Currently Free" on page 232

Request Blocks Currently Free

Element identifier rb_free

Element type gauge

Table 164. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description
This element indicates the number of request blocks currently free.

Usage Use the number of request blocks currently free in conjunction with the *fcm_num_rqb* configuration parameter to determine the current request block utilization. You can use this information to tune *fcm_num_rqb*.

Minimum Request Blocks

Element identifier rb_free_bottom

Element type water mark

Table 165. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	fcm	Basic

Description
The lowest number of free request blocks reached during processing.

Usage Use this element in conjunction with the *fcm_num_rqb* configuration parameter to determine the maximum request block utilization. If rb_free_bottom is low, you should increase *fcm_num_rqb* to ensure that operations do not run out of request blocks. If rb_free_bottom is high, you can decrease *fcm_num_rqb* to conserve system resources.

Related reference:

- "Request Blocks Currently Free" on page 233

Connection Status

Element identifier connection_status

Element type information

Table 166. 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 FCM Buffers Sent" on page 235
- "Total FCM Buffers Received" on page 235

Total FCM Buffers Sent

Element identifier total_buffers_sent

Element type counter

Table 167. 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:

- "Connection Status" on page 234
- "Total FCM Buffers Received" on page 235

Total FCM Buffers Received

Element identifier total_buffers_rcvd

Element type counter

Table 168. 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" on page 234
- "Total FCM Buffers Sent" on page 235

Database configuration

Database configuration monitor elements

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

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

- 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 the main issue when trying to improve the performance of your server And so, proper configuration of the buffer pools are probably the most important consideration 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. That is, 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 - ((pool_data_p_reads + pool_index_p_reads) /
  (pool_data_l_reads + pool_index_l_reads))) * 100%
```

This calculation takes into account all of the pages (index and data) that are cached by the buffer pool.

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 increasing its size. But 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 - ((pool_index_p_reads) / (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.

- Buffer Pool Data Logical Reads
- Buffer Pool Data Physical Reads
- Buffer Pool Data Writes
- Buffer Pool Index Logical Reads
- Buffer Pool Index Physical Reads
- Buffer Pool Index Writes
- Total Buffer Pool Physical Read Time
- Total Buffer Pool Physical Write Time
- Database Files Closed
- Buffer Pool Asynchronous Data Reads
- Buffer Pool Asynchronous Data Writes
- Buffer Pool Asynchronous Index Writes
- Buffer Pool Asynchronous Index Reads
- Buffer Pool Asynchronous Read Time
- Buffer Pool Asynchronous Write Time
- Buffer Pool Asynchronous Read Requests
- Buffer Pool Log Space Cleaners Triggered
- Buffer Pool Victim Page Cleaners Triggered
- Buffer Pool Threshold Cleaners Triggered
- Buffer Pool Information
- Buffer Pool Name
- Time Waited for Prefetch
- Extended storage data elements

- Total Number of Pages Read by Vectored IO
- Number of Block IO Requests
- Number of Vectored IO Requests
- Total Number of Pages Read by Block IO
- Number of Physical Page Maps

Buffer Pool Data Logical Reads

Element identifier pool_data_l_reads

Element type counter

Table 169. 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 170. Event Monitoring Information

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

Description

Indicates the number of logical read requests for data pages that have gone through the buffer pool.

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 - (pool_data_p_reads / pool_data_l_reads)
```

In conjunction with pool_data_p_reads, pool_index_p_reads, and pool_index_l_reads, you can calculate the overall buffer pool hit ratio using the following formula:

```
1 - ((pool_data_p_reads + pool_index_p_reads)
    / (pool_data_l_reads + pool_index_l_reads))
```

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 reference:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Data Physical Reads" on page 240
- "Buffer Pool Data Writes" on page 241
- "Buffer Pool Index Logical Reads" on page 243
- "Buffer Pool Index Physical Reads" on page 244

Buffer Pool Data Physical Reads

Element identifier pool_data_p_reads

Element type counter

Table 171. 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 172. Event Monitoring Information

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

Description

The number of read requests that required I/O to get data pages into the buffer pool.

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

Related reference:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Data Logical Reads" on page 239
- "Buffer Pool Data Writes" on page 241
- "Buffer Pool Index Logical Reads" on page 243
- "Buffer Pool Index Physical Reads" on page 244
- "Buffer Pool Asynchronous Data Reads" on page 249

Buffer Pool Data Writes

Element identifier pool_data_writes

Element type counter

Table 173. 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 174. 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.

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:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Data Logical Reads" on page 239
- "Buffer Pool Data Physical Reads" on page 240
- "Total Buffer Pool Physical Write Time" on page 247

• "Buffer Pool Asynchronous Data Writes" on page 250

Buffer Pool Index Logical Reads

Element identifier pool_index_l_reads

Element type counter

Table 175. 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 176. Event Monitoring Information

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

Description

Indicates the number of logical read requests for index pages that have gone through the buffer pool.

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 - (pool_index_p_reads / 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:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Data Logical Reads" on page 239
- "Buffer Pool Data Physical Reads" on page 240
- "Buffer Pool Data Writes" on page 241
- "Buffer Pool Index Physical Reads" on page 244
- "Buffer Pool Index Writes" on page 245

Buffer Pool Index Physical Reads

Element identifier pool_index_p_reads

Element type counter

Table 177. 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 178. Event Monitoring Information

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

Description

Indicates the number of physical read requests to get index pages into the buffer pool.

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

Related reference:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Data Logical Reads" on page 239
- "Buffer Pool Data Physical Reads" on page 240
- "Buffer Pool Index Logical Reads" on page 243
- "Buffer Pool Index Writes" on page 245

Buffer Pool Index Writes

Element identifier pool_index_writes

Element type counter

Table 179. 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 180. 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:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Index Logical Reads" on page 243
- "Buffer Pool Index Physical Reads" on page 244
- "Buffer Pool Asynchronous Index Writes" on page 251

Total Buffer Pool Physical Read Time

Element identifier pool_read_time

Element type counter

Table 181. 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 182. 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 elapsed time spent processing read requests that caused data or index pages to be physically read from disk to buffer pool.

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:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Data Physical Reads" on page 240
- "Buffer Pool Index Physical Reads" on page 244
- "Buffer Pool Asynchronous Read Time" on page 253

Total Buffer Pool Physical Write Time

Element identifier pool_write_time

Element type counter

Table 183. 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 184. 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.

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:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Buffer Pool Data Writes" on page 241
- "Buffer Pool Index Writes" on page 245

Database Files Closed

Element identifier files_closed

Element type counter

Table 185. 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 186. 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).

Buffer Pool Asynchronous Data Reads

Element identifier pool_async_data_reads

Element type counter

Table 187. 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 188. Event Monitoring Information

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

Description
The number of pages read asynchronously into the buffer pool.

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:

```
pool_data_p_reads - 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:

- "Buffer Pool Data Physical Reads" on page 240
- "Buffer Pool Asynchronous Read Time" on page 253
- "Buffer Pool Asynchronous Read Requests" on page 255
- "Direct Reads From Database" on page 267

Buffer Pool Asynchronous Data Writes

Element identifier pool_async_data_writes

Element type counter

Table 189. 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 190. 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:

```
pool_data_writes - pool_async_data_writes
```

By comparing the ratio of asynchronous to synchronous writes, you can gain insight into how well the buffer pool page cleaners are performing. This ratio can be helpful when you are tuning the *num_iocleaners* configuration parameter.

For more information about asynchronous page cleaners, see the *Administration Guide*.

Related reference:

- "Buffer Pool Data Writes" on page 241
- "Buffer Pool Asynchronous Index Writes" on page 251
- "Buffer Pool Asynchronous Write Time" on page 254
- "Buffer Pool Log Space Cleaners Triggered" on page 256
- "Buffer Pool Victim Page Cleaners Triggered" on page 256
- "Buffer Pool Threshold Cleaners Triggered" on page 257
- "Direct Writes to Database" on page 268

Buffer Pool Asynchronous Index Writes

Element identifier pool_async_index_writes

Element type counter

Table 191. 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 192. 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:

```
pool_index_writes - 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:

- "Buffer Pool Index Writes" on page 245
- "Buffer Pool Asynchronous Data Writes" on page 250
- "Buffer Pool Asynchronous Index Reads" on page 252
- "Buffer Pool Asynchronous Write Time" on page 254
- "Buffer Pool Log Space Cleaners Triggered" on page 256
- "Buffer Pool Victim Page Cleaners Triggered" on page 256
- "Buffer Pool Threshold Cleaners Triggered" on page 257
- "Direct Writes to Database" on page 268

Buffer Pool Asynchronous Index Reads

Element identifier pool_async_index_reads

Element type counter

Table 193. 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 194. Event Monitoring Information

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

Description

The number of index pages read asynchronously into the buffer pool by a prefetcher.

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:

```
pool_index_p_reads - 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:

- "Buffer Pool Index Physical Reads" on page 244
- "Buffer Pool Asynchronous Data Writes" on page 250
- "Buffer Pool Asynchronous Index Writes" on page 251
- "Buffer Pool Asynchronous Read Time" on page 253
- "Buffer Pool Log Space Cleaners Triggered" on page 256
- "Buffer Pool Victim Page Cleaners Triggered" on page 256
- "Buffer Pool Threshold Cleaners Triggered" on page 257
- "Direct Reads From Database" on page 267

Buffer Pool Asynchronous Read Time

Element identifier pool_async_read_time

Element type counter

Table 195. 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 196. Event Monitoring Information

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

Description

The total elapsed time spent reading by database manager prefetchers.

Usage You can use this element to calculate the elapsed time for synchronous reading, using the following formula:

```
pool_read_time - pool_async_read_time
```

You can also use this element to calculate the average asynchronous read time using the following formula:

```
pool_async_read_time / pool_async_data_reads
```

These calculations can be used to understand the I/O work being performed.

Related reference:

- "Total Buffer Pool Physical Read Time" on page 246
- "Buffer Pool Asynchronous Data Reads" on page 249
- "Buffer Pool Asynchronous Read Requests" on page 255
- "Direct Read Time" on page 270

Buffer Pool Asynchronous Write Time

Element identifier pool_async_write_time

Element type counter

Table 197. Snapshot Monitoring Information

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

For snapshot monitoring, this counter can be reset.

Table 198. Event Monitoring Information

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

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:

```
pool_write_time - pool_async_write_time
```

You can also use this element to calculate the average asynchronous read time using the following formula:

```
pool_async_write_time
 / (pool_async_data_writes
   + pool_async_index_writes)
```

These calculations can be used to understand the I/O work being performed.

Related reference:

- "Total Buffer Pool Physical Write Time" on page 247
- "Buffer Pool Asynchronous Data Writes" on page 250
- "Buffer Pool Asynchronous Index Writes" on page 251
- "Buffer Pool Asynchronous Read Requests" on page 255
- "Direct Write Time" on page 271

Buffer Pool Asynchronous Read Requests

Element identifier pool_async_data_read_reqs

Element type counter

Table 199. Snapshot Monitoring Information

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

For snapshot monitoring, this counter can be reset.

Table 200. Event Monitoring Information

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

Description

The number of asynchronous read requests.

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

```
pool_async_data_reads / 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:

- "Buffer Pool Asynchronous Data Reads" on page 249

Buffer Pool Log Space Cleaners Triggered

Element identifier pool_lsn_gap_clns

Element type counter

Table 201. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 202. 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.

Related reference:

- "Buffer Pool Victim Page Cleaners Triggered" on page 256
- "Buffer Pool Threshold Cleaners Triggered" on page 257

Buffer Pool Victim Page Cleaners Triggered

Element identifier pool_drty_pg_steal_clns

Element type counter

Table 203. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 204. 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:

```
  pool_drty_pg_steal_clns
/ ( pool_drty_pg_steal_clns
  + pool_drty_pg_thrsh_clns
  + pool_lsn_gap_clns)
```

If this ratio is low, it may indicate that you have defined too many page cleaners. If your *chngpgs_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*).

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:

- "Buffer Pool Log Space Cleaners Triggered" on page 256
- "Buffer Pool Threshold Cleaners Triggered" on page 257

Buffer Pool Threshold Cleaners Triggered

Element identifier pool_drty_pg_thrsh_clns

Element type counter

Table 205. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool

For snapshot monitoring, this counter can be reset.

Table 206. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

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 *chngpgs_thresh* configuration parameter. It is a percentage applied to the buffer pool size. When the number of dirty pages in the pool exceeds this value, the cleaners are triggered.

If this value is set too low, pages might be written out too early, requiring them to be read back in. If set too high, then too many pages may accumulate, requiring users to write out pages synchronously. See the *Administration Guide* for more information.

Related reference:

- "Buffer Pool Log Space Cleaners Triggered" on page 256

Buffer Pool Name

Element identifier bp_name

Element type information

Table 207. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Basic

Description

The name of the buffer pool.

Usage A new database has a default buffer pool called IBMDEFAULTBP with a size determined by the platform. Each database requires at least one buffer pool. However, 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.

Time Waited for Prefetch

Element identifier prefetch_wait_time

Element type counter

Table 208. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Buffer Pool
Application	appl	Buffer Pool

Table 209. 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.

Number of Vectored IO Requests

Element identifier vectored_ios

Element type gauge

Table 210. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

The number of vectored I/O requests.

Usage Use this element to determine how often vectored I/O is being done.

Total Number of Pages Read by Vectored IO

Element identifier pages_from_vectored_ios

Element type gauge

Table 211. 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.

Number of Block IO Requests

Element identifier block_ios

Element type gauge

Table 212. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description
The number of block I/O requests.

Usage If block-based buffer pool is enabled, this element will report how often block I/O is being done. Otherwise this element will return 0. If block-based buffer pool is enabled and this number is very high, or very low, consider changing the block size. For example, if extent size is 2 and 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.

Total Number of Pages Read by Block IO

Element identifier pages_from_block_ios

Element type gauge

Table 213. 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.

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. It will help determine if a change to the block size is necessary.

Number of Physical Page Maps

Element identifier physical_page_maps

Element type gauge

Table 214. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

The number of physical page maps.

Extended storage

Extended storage monitor elements: Extended storage provides a secondary level of storage for buffer pools. This allows a user to access memory beyond the maximum allowed for each process. Extended storage consists of segments that will be allocated in addition to the buffer pools. The extended storage will assign pages to segments that are attached or detached, as needed. The number and size of segments are configurable. Attachment is allowed to only one segment at a given time.

There is one extended storage for all buffer pools, and each buffer pool can be configured to use it or not.

Extended storage should only be used on systems with very large amount of real memory. These are systems that have more memory than can be attached to by a single process.

If you have extended storage set on for a buffer pool, all pages removed from the buffer pool will be written to extended storage. Each of these writes has a cost associated with it. Some of these pages may never be required or they may be forced out of extended storage before they are ever read back into the buffer pool.

You can calculate the extended storage read/write ratio as follows:

```
(data + index copied from extended storage)
/ (data + index copied to extended storage)
```

Where the numerator in this equation is pages from extended storage to buffer pool and the denominator is pages from buffer pool to extended storage.

The top portion of this equation represents a performance saving. When a page is transferred from extended storage to buffer pool, you save a system I/O call. However, you still incur the cost of attaching to the extended memory segment, copying the page, and detaching from the segment. The bottom part represents the cost of transferring a page to extended storage, that is, attaching to the segment, copying the page, and detaching.

The higher the ratio, the more likely you are to benefit from extended storage. In general, extended storage is particularly useful if I/O activity is very high on your system.

There is a crossover point where the cost of copying pages to be removed from the buffer pool to extended storage equals the savings from reading pages from extended storage, instead of having to read them from disk. This crossover point is affected by:
- cost of an I/O on your system
- cost of copying data in memory and accessing shared memory segments

It is difficult to establish an exact crossover point. To establish a baseline, you must experiment by enabling extended storage for different buffer pools, and determine whether it improves your overall database performance. This can be measured by using application benchmarks. For instance, you may want to monitor transaction rates and execution time.

Once you have established that extended storage is beneficial for some buffer pools. You want to measure the read/write ratio to obtain a baseline. This ratio is most important during database creation and initial setup. After that, you want to monitor this ratio to ensure that it is not deviating from the initial baseline.

The following elements provide information about buffer pools and extended storage.
- Buffer Pool Data Pages to Extended Storage
- Buffer Pool Index Pages to Extended Storage
- Buffer Pool Data Pages from Extended Storage
- Buffer Pool Index Pages from Extended Storage

Buffer Pool Data Pages to Extended Storage:

Element identifier pool_data_to_estore

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	-
Connection	event_conn	-
Tablespaces	event_tablespace	-

Description

Number of buffer pool data pages copied to extended storage.

Usage Pages are copied from the buffer pool to extended storage, when they are selected as victim pages. This copying is required to make space for new pages in the buffer pool.

Related reference:

- "Buffer Pool Index Pages to Extended Storage" on page 263
- "Buffer Pool Data Pages from Extended Storage" on page 264
- "Buffer Pool Index Pages from Extended Storage" on page 264

Buffer Pool Index Pages to Extended Storage:

Element identifier pool_index_to_estore

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

For snapshot monitoring, this counter can be reset.

Table 218. Event Monitoring Information

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

Description

Number of buffer pool index pages copied to extended storage.

Usage Pages are copied from the buffer pool to extended storage, when they are selected as victim pages. This copying is required to make space for new pages in the buffer pool.

Related reference:

- "Buffer Pool Data Pages to Extended Storage" on page 262
- "Buffer Pool Data Pages from Extended Storage" on page 264
- "Buffer Pool Index Pages from Extended Storage" on page 264

Buffer Pool Data Pages from Extended Storage:

Element identifier pool_data_from_estore

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

For snapshot monitoring, this counter can be reset.

Table 220. Event Monitoring Information

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

Description

Number of buffer pool data pages copied from extended storage.

Usage Required pages are copied from extended storage to the buffer pool, if they are not in the buffer pool, but are in extended storage. This copying may incur the cost of connecting to the shared memory segment, but saves the cost of a disk read.

Related reference:

- "Buffer Pool Data Pages to Extended Storage" on page 262
- "Buffer Pool Index Pages to Extended Storage" on page 263
- "Buffer Pool Index Pages from Extended Storage" on page 264

Buffer Pool Index Pages from Extended Storage:

Element identifier pool_index_from_estore

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

For snapshot monitoring, this counter can be reset.

Table 222. Event Monitoring Information

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

Description

Number of buffer pool index pages copied from extended storage.

Usage Required index pages are copied from extended storage to the buffer pool, if they are not in the buffer pool, but are in extended storage. This copying may incur the cost of connecting to the shared memory segment, but saves the cost of a disk read.

Related reference:

- "Buffer Pool Data Pages to Extended Storage" on page 262
- "Buffer Pool Index Pages to Extended Storage" on page 263
- "Buffer Pool Data Pages from Extended Storage" on page 264

Dynamic buffer pool

Dynamic buffer pool monitor elements: The following elements provide information about dynamic buffer pools.

- Current Size of Buffer Pool
- New Buffer Pool Size
- Number of Pages Left to Remove
- Number of Table Spaces Mapped to Buffer Pool

Current Size of Buffer Pool:

Element identifier bp_cur_buffsz

Element type gauge

Table 223. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	Buffer Pool

Description

Current buffer pool size.

New Buffer Pool Size:

Element identifier bp_new_buffsz

Element type information

Table 224. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	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.

Number of Pages Left to Remove:

Element identifier bp_pages_left_to_remove

Element type gauge

Table 225. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	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.

Number of Table Spaces Mapped to Buffer Pool:

Element identifier bp_tbsp_use_count

Element type gauge

Table 226. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Buffer Pool	bufferpool	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 From Database
- Direct Writes to Database
- Direct Read Requests
- Direct Write Requests
- Direct Read Time
- Direct Write Time

Direct Reads From Database

Element identifier direct_reads

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	-
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:

```
direct_reads / direct_read_reqs
```

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

Direct reads are performed in units, the smallest being a 512-byte sector. They are used when:
- Reading LONG VARCHAR columns
- Reading LOB (large object) columns
- Performing a backup

Related reference:
- "Direct Writes to Database" on page 268
- "Direct Read Requests" on page 269
- "Direct Read Time" on page 270

Direct Writes to Database

Element identifier direct_writes

Element type counter

Table 229. Snapshot Monitoring Information

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

For snapshot monitoring, this counter can be reset.

Table 230. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
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.

```
direct_writes / direct_write_reqs
```

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

Direct writes are performed in units, the smallest being a 512-byte sector. They are used when:
- Writing LONG VARCHAR columns
- Writing LOB (large object) columns
- Performing a restore
- Performing a load.

Related reference:
- "Direct Reads From Database" on page 267
- "Direct Write Requests" on page 270
- "Direct Write Time" on page 271

Direct Read Requests

Element identifier direct_read_reqs

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
Application	appl	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	-
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:

```
direct_reads / direct_read_reqs
```

Related reference:

- "Direct Reads From Database" on page 267
- "Direct Write Requests" on page 270
- "Direct Read Time" on page 270

Direct Write Requests

Element identifier direct_write_reqs

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
Application	appl	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	-
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:

```
direct_writes / direct_write_reqs
```

Related reference:

- "Direct Writes to Database" on page 268
- "Direct Read Requests" on page 269
- "Direct Write Time" on page 271

Direct Read Time

Element identifier direct_read_time

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
Application	appl	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	-
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:

```
direct_read_time / direct_reads
```

A high average time may indicate an I/O conflict.

Related reference:

- "Direct Reads From Database" on page 267
- "Direct Read Requests" on page 269
- "Direct Write Time" on page 271

Direct Write Time

Element identifier direct_write_time

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

Table 237. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	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	-
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:

```
direct_write_time / direct_writes
```

A high average time may indicate an I/O conflict.

Related reference:

- "Direct Writes to Database" on page 268
- "Direct Write Requests" on page 270
- "Direct Read Time" on page 270

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:
- Catalog Cache Lookups
- Catalog Cache Inserts
- Catalog Cache Overflows
- Catalog Cache High Water Mark

Catalog Cache Lookups

Element identifier cat_cache_lookups

Element type counter

Table 239. Snapshot Monitoring Information

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

For snapshot monitoring, this counter can be reset.

Table 240. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-
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 - (cat_cache_inserts / 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:

- "Catalog Cache Inserts" on page 274
- "Catalog Cache Overflows" on page 275
- "Data Definition Language (DDL) SQL Statements" on page 378
- "Catalog Cache High Water Mark" on page 276

Catalog Cache Inserts

Element identifier cat_cache_inserts

Element type counter

Table 241. 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 242. 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 - (Catalog Cache Inserts / Catalog Cache Lookups)
```

See cat_cache_lookups for more information on using this element.

Related reference:

- “Catalog Cache Lookups” on page 273
- “Catalog Cache Overflows” on page 275
- “Catalog Cache High Water Mark” on page 276

Catalog Cache Overflows

Element identifier cat_cache_overflows

Element type counter

Table 243. 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 244. 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. Overflows of the catalog cache may cause unnecessary lock escalations, resulting in loss of concurrency, or out of memory errors from other heaps allocated out of the database shared memory. As well, overflows of the catalog cache may result in performance degradation.

Catalog cache space is reclaimed by evicting table descriptor information for tables, views, or aliases and/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:

- "Catalog Cache Lookups" on page 273
- "Catalog Cache Inserts" on page 274
- "Catalog Cache High Water Mark" on page 276

Catalog Cache High Water Mark

Element identifier cat_cache_size_top

Element type watermark

Table 245. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 246. 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.

When the catalog cache overflows, memory is temporarily borrowed from other entities in database shared memory (for example, lock list, database heap or package cache). This can result in memory shortage errors from these entities or performance degradation from

concurrency reduction due to unnecessary lock escalations. 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:

- "Catalog Cache Overflows" on page 275

Package cache

Package cache monitor elements

When a new row is being inserted or an existing row is being updated, resulting in an increased record size, the page where this record is to be placed may have enough free space, but that space could be fragmented. In these cases the page may require reorganization, which moves all fragmented space to a contiguous area, where the new record can be written. Such a page reorganization (page reorg) is very expensive to perform.

- Package Cache Lookups
- Package Cache Inserts
- Package Cache Overflows
- Package Cache High Water Mark

Package Cache Lookups

Element identifier pkg_cache_lookups

Element type counter

Table 247. 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 248. 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 - (Package Cache Inserts / Package Cache Lookups)
```

The package cache hit ratio tells you whether or not the package cache is being used effectively. If the hit ratio is high (more than 0.8), the cache is performing well. A smaller 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 and not 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:

- "Package Cache Inserts" on page 279
- "Static SQL Statements Attempted" on page 371
- "Dynamic SQL Statements Attempted" on page 372
- "Data Definition Language (DDL) SQL Statements" on page 378

Package Cache Inserts

Element identifier pkg_cache_inserts

Element type counter

Table 249. 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 250. 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 - (Package Cache Inserts / Package Cache Lookups)
```

See pkg_cache_lookups for information on using this element.

Related reference:

- “Package Cache Lookups” on page 277

Package Cache Overflows

Element identifier pkg_cache_num_overflows

Element type counter

Table 251. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

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 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. Overflows of the package cache can cause unnecessary lock escalations, resulting in loss of concurrency, or out of memory errors from the other heaps allocated out of the database shared memory, as well as performance degradation.

Related reference:

- “Package Cache Inserts” on page 279
- “Static SQL Statements Attempted” on page 371

- "Dynamic SQL Statements Attempted" on page 372
- "Data Definition Language (DDL) SQL Statements" on page 378

Package Cache High Water Mark

Element identifier pkg_cache_size_top

Element type water mark

Table 253. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 254. 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.

When the package cache overflows, memory is temporarily borrowed from other entities in database shared memory (for example, lock list, database heap, or catalog cache). This can result in memory shortage errors from these entities or performance degradation from concurrency reduction due to unnecessary lock escalations. You can determine the minimum size of the package cache required by your workload by:

```
maximum package cache size / 4096
```

Rounding the result up to a whole number, indicates the minimum number of 4K pages required by the package cache to avoid overflow.

Related reference:

- "Package Cache Overflows" on page 280

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:

- Maximum Shared Workspace Size
- Shared Workspace Overflows
- Shared Workspace Section Lookups
- Shared Workspace Section Inserts
- Maximum Private Workspace Size
- Private Workspace Overflows
- Private Workspace Section Lookups
- Private Workspace Section Inserts

Maximum Shared Workspace Size

Element identifier shr_workspace_size_top

Element type water mark

Table 255. Snapshot Monitoring Information

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

Table 256. 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 APPL_CTL_HEAP_SZ.

Related reference:

- "Shared Workspace Overflows" on page 283

Shared Workspace Overflows

Element identifier shr_workspace_num_overflows

Element type counter

Table 257. 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 258. 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:

- "Maximum Shared Workspace Size" on page 282

Shared Workspace Section Lookups

Element identifier shr_workspace_section_lookups

Element type counter

Table 259. 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 260. 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:

- "Shared Workspace Section Inserts" on page 285

Shared Workspace Section Inserts

Element identifier shr_workspace_section_inserts

Element type counter

Table 261. 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 262. 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:

• "Shared Workspace Section Lookups" on page 284

Maximum Private Workspace Size

Element identifier priv_workspace_size_top

Element type water mark

Table 263. Snapshot Monitoring Information

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

Table 264. 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:

• "Private Workspace Overflows" on page 286

Private Workspace Overflows

Element identifier priv_workspace_num_overflows

Element type counter

Table 265. 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 266. 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:

• "Maximum Private Workspace Size" on page 285

Private Workspace Section Lookups

Element identifier priv_workspace_section_lookups

Element type counter

Table 267. 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 268. 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:

- "Private Workspace Section Inserts" on page 288

Private Workspace Section Inserts

Element identifier priv_workspace_section_inserts

Element type counter

Table 269. 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 270. 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:

• "Private Workspace Section Lookups" on page 287

Database heap

Database heap monitor elements

The following database system monitor elements are used for database heaps:

- Maximum Database Heap Allocated

Maximum Database Heap Allocated

Element identifier db_heap_top

Element type water mark

Table 271. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 272. 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:

- Maximum Secondary Log Space Used
- Maximum Total Log Space Used
- Secondary Logs Allocated Currently
- Number of Log Pages Read
- Number of Log Pages Written
- Unit of Work Log Space Used
- Total Log Space Used
- Total Log Available

Maximum Secondary Log Space Used

Element identifier sec_log_used_top

Element type water mark

Table 273. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 274. 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:

- logfilsiz
- logprimary
- logsecond
- logretain

The value will be zero if the database does not have any secondary log files. This would be the case if there were none defined.

For more information, see the *Administration Guide*.

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Related reference:

- "Maximum Total Log Space Used" on page 290
- "Secondary Logs Allocated Currently" on page 292
- "Unit of Work Log Space Used" on page 294

Maximum Total Log Space Used

Element identifier tot_log_used_top

Element type water mark

Table 275. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 276. 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 you 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:

```
logprimary x logfilsiz x 4096 (see note below)
```

You can use this element in conjunction with *sec_log_used_top* and *sec_logs_allocated* to show your current dependency on secondary logs.

This value includes space used in both primary and secondary log files.

As a result, you may need to adjust the following configuration parameters:

- logfilsiz
- logprimary
- logsecond
- logretain

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:

- "Maximum Secondary Log Space Used" on page 289
- "Secondary Logs Allocated Currently" on page 292
- "Unit of Work Log Space Used" on page 294

Secondary Logs Allocated Currently

Element identifier sec_logs_allocated

Element type gauge

Table 277. 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:

- logfilsiz
- logprimary
- logsecond
- logretain

For more information, see the *Administration Guide*.

Related reference:

- "Maximum Secondary Log Space Used" on page 289
- "Maximum Total Log Space Used" on page 290
- "Unit of Work Log Space Used" on page 294

Number of Log Pages Read

Element identifier log_reads

Element type counter

Table 278. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 279. 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:

- "Number of Log Pages Written" on page 293

Number of Log Pages Written

Element identifier log_writes

Element type counter

Table 280. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

For snapshot monitoring, this counter can be reset.

Table 281. 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:

- "Number of Log Pages Read" on page 292

Unit of Work Log Space Used

Element identifier uow_log_space_used

Element type gauge

Table 282. Snapshot Monitoring Information

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

Table 283. 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:

- "Maximum Secondary Log Space Used" on page 289
- "Maximum Total Log Space Used" on page 290
- "Secondary Logs Allocated Currently" on page 292

Total Log Space Used

Element identifier total_log_used

Element type gauge

Table 284. 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:

- 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:

- "Maximum Total Log Space Used" on page 290
- "Secondary Logs Allocated Currently" on page 292
- "Unit of Work Log Space Used" on page 294
- "Application with Oldest Transaction" on page 174

Total Log Available

Element identifier total_log_available

Element type gauge

Table 285. 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:

- logfilsiz
- logprimary
- logsecond

If total_log_available goes down to 0, SQL0964N will be returned. You may need to increase the above configuration parameters, or end the oldest transaction by COMMIT, ROLLBACK or FORCE APPLICATION.

If logsecond is set to -1 this element will contain SQLM_LOGSPACE_INFINITE.

Note: While the database system monitor information is given in bytes, the configuration parameters are set in pages, which are each 4K bytes.

Related reference:

Database configuration monitor elements

- "Secondary Logs Allocated Currently" on page 292
- "Unit of Work Log Space Used" on page 294
- "Total Log Space Used" on page 294
- "Application with Oldest Transaction" on page 174

Database and application activity

Database and application activity monitor elements

The following sections provide information on database and application activity.

- Locks and deadlocks data elements
- Lock wait information data elements
- Rollforward monitoring data elements
- Table activity data elements
- Table reorganization data elements
- SQL cursors data elements
- SQL statement activity data elements
- SQL statement details data elements
- Subsection details data elements
- Dynamic SQL data elements
- Intra-query parallelism data elements
- CPU usage data elements
- Snapshot monitoring data elements
- Event monitoring data elements

Locks and deadlocks

Locks and deadlocks monitor elements

The following elements provide information about locks and deadlocks:

- Locks Held
- Total Lock List Memory In Use
- Deadlocks Detected
- Number of Lock Escalations
- Exclusive Lock Escalations
- Lock Mode
- Lock Status
- Lock Object Type Waited On
- Lock Object Name
- Lock Node

- Number of Lock Timeouts
- Maximum Number of Locks Held
- Connections Involved in Deadlock
- Lock Escalation
- Lock Mode Requested
- Deadlock Event Identifier
- Partition Number Where Deadlock Occurred
- Participant within Deadlock
- Participant Holding a Lock on the Object Required by Application
- Rolled Back Application Participant
- Number of Locks Reported
- Lock Name
- Lock Attributes
- Lock Release Flags
- Lock Count
- Lock Hold Count
- Original Lock Mode Before Conversion

Locks Held

Element identifier locks_held

Element type gauge

Table 286. 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 287. 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:

- "Number of Lock Escalations" on page 299
- "Exclusive Lock Escalations" on page 301
- "Maximum Number of Locks Held" on page 307

Total Lock List Memory In Use

Element identifier lock_list_in_use

Element type gauge

Table 288. 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 Detected

Element identifier deadlocks

Element type counter

Table 289. 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 290. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_db	-

Table 290. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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.

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:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Number of Lock Escalations" on page 299
- "Exclusive Lock Escalations" on page 301
- "Application ID Holding Lock" on page 320

Number of Lock Escalations

Element identifier lock_escals

Element type counter

Table 291. 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 292. 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
- 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:

- "Database Activation Timestamp" on page 164
- "Exclusive Lock Escalations" on page 301
- "Maximum Number of Locks Held" on page 307

Exclusive Lock Escalations

Element identifier x_lock_escals

Element type counter

Table 293. 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 294. 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 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:

- "Database Activation Timestamp" on page 164
- "Connection Request Start Timestamp" on page 191
- "Number of Lock Escalations" on page 299
- "Maximum Number of Locks Held" on page 307

Lock Mode

Element identifier lock_mode

Element type information

Table 295. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	lock	Lock
Lock	lock_wait	Lock

Table 296. 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).

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

Element identifier lock_status

Element type information

Table 297. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic

Table 298. 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" on page 302
- "Lock Object Type Waited On" on page 304
- "Lock Object Name" on page 305
- "Table File ID" on page 359

Lock Object Type Waited On

Element identifier lock_object_type

Element type information

Table 299. 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 300. 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 objects may define one of the following types:

- Table space (SQLM_TABLESPACE_LOCK in sqlmon.h)
- Table
- Block
- Record (or row)
- Internal (another type of lock held internally by the database manager).

Lock Object Name

Element identifier lock_object_name

Element type information

Table 301. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	appl_lock_list	Lock
Lock	lock	Basic

Table 302. 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 It is the name of the object for table-level locks 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.

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 Waited On" on page 304
- "Table Space Name" on page 326
- "Table Name" on page 349
- "Table Schema Name" on page 350

Lock Node

Element identifier lock_node

Element type information

Table 303. 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.

Number of Lock Timeouts

Element identifier lock_timeouts

Element type counter

Table 304. 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 305. 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.

Maximum Number of Locks Held

Element identifier locks_held_top

Element type counter

Table 306. 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:

```
(locklist * 4096 / 36 ) * (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" on page 297
- "Number of Lock Escalations" on page 299
- "Exclusive Lock Escalations" on page 301

Connections Involved in Deadlock

Element identifier dl_conns

Element type gauge

Table 307. 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

Element identifier lock_escalation

Element type information

Table 308. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Lock
Lock	lock_wait	Lock

Table 309. 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

Element identifier lock_mode_requested

Element type information

Table 310. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock_wait	Lock

Table 311. 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 Event Identifier

Element identifier deadlock_id

Element type information

Table 312. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

Description
The deadlock identifier for a deadlock.

Usage Use this element in your monitoring application to correlate deadlock connection event records with deadlock event records.

Partition Number Where Deadlock Occurred

Element identifier deadlock_node

Element type information

Table 313. 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 within Deadlock

Element identifier participant_no

Element type information

Table 314. 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 Holding a Lock on the Object Required by Application

Element identifier participant_no_holding_lk

Element type information

Table 315. 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:

- "Application ID Holding Lock" on page 320

Rolled Back Application Participant

Element identifier rolled_back_participant_no

Element type information

Table 316. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	event_deadlock	-

Description

The participant number identifying the rolled back application.

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 Application" on page 321

Number of Locks Reported

Element identifier locks_in_list

Element type information

Table 317. 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

Element identifier lock_name

Element type information

Table 318. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	lock_wait

Table 319. 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

Element identifier lock_attributes

Element type information

Table 320. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	Basic

Table 321. 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_RR_IN_BLOCK	RR lock ″in″ block.
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

Element identifier lock_release_flags

Element type information

Table 322. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	Basic

Table 323. 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.

Lock Count

Element identifier lock_count

Element type gauge

Table 324. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic

Table 325. 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

Element identifier lock_hold_count

Element type gauge

Table 326. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic

Table 327. 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.

Original Lock Mode Before Conversion

Element identifier lock_current_mode

Element type information

Table 328. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Lock	lock	Basic
Lock	lock_wait	Basic

Table 329. 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.

Lock wait information

Lock wait information monitor elements

The following elements provide information is returned when a DB2 agent working on behalf of an application is waiting to obtain a lock:

- Lock Waits
- Time Waited On Locks
- Current Agents Waiting On Locks
- Total Time Unit of Work Waited on Locks
- Lock Wait Start Timestamp
- Agent ID Holding Lock
- Application ID Holding Lock

- Sequence Number Holding Lock
- Rolled Back Application
- Rolled Back Agent
- Rolled Back Sequence Number

Lock Waits

Element identifier lock_waits

Element type counter

Table 330. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Lock
Application	appl	Lock

For snapshot monitoring, this counter can be reset.

Table 331. 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. See the *Administration Guide* for more information.

Related reference:

• "Connection Request Start Timestamp" on page 191

• "Time Waited On Locks" on page 317

Time Waited On Locks

Element identifier lock_wait_time

Element type counter

Table 332. 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 333. 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.

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.

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:

- "Lock Waits" on page 316
- "Current Agents Waiting On Locks" on page 318

Current Agents Waiting On Locks

Element identifier locks_waiting

Element type gauge

Table 334. 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:

- "Applications Connected Currently" on page 204

Total Time Unit of Work Waited on Locks

Element identifier uow_lock_wait_time

Element type counter

Table 335. 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 Timestamp

Element identifier lock_wait_start_time

Element type timestamp

Table 336. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock, Timestamp
Lock	lock_wait	Lock, Timestamp

Table 337. 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" on page 319

Agent ID Holding Lock

Element identifier agent_id_holding_lock

Element type information

Table 338. 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.

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 Timestamp" on page 319
- "Application ID Holding Lock" on page 320

Application ID Holding Lock

Element identifier appl_id_holding_lk

Element type information

Table 339. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Lock	appl_lock_list	Lock
Lock	lock_wait	Lock

Table 340. 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:

- "Deadlocks Detected" on page 298
- "Agent ID Holding Lock" on page 319

Sequence Number Holding Lock

Element identifier sequence_no_holding_lk

Element type information

Table 341. Snapshot Monitoring Information

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

Table 342. 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 Application

Element identifier rolled_back_appl_id

Element type information

Table 343. 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:

- "Maximum Number of Coordinating Agents" on page 209

Rolled Back Agent

Element identifier rolled_back_agent_id

Element type information

Table 344. 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:

- "Maximum Number of Coordinating Agents" on page 209

Rolled Back Sequence Number

Element identifier rolled_back_sequence_no

Element type information

Table 345. 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:

- Rollforward Timestamp
- Tablespace Being Rolled Forward
- Rollforward Type
- Log Being Rolled Forward
- Log Phase
- Number of Rollforward Table Spaces

Rollforward Timestamp

Element identifier rf_timestamp

Element type timestamp

Table 346. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	rollforward	Timestamp

Description
The timestamp of the log being processed.

Usage If a rollforward is in progress, this is the timestamp of the log record being processed. This is an indicator of the data changes that will be recovered.

Related reference:

- "Tablespace Being Rolled Forward" on page 323

Tablespace Being Rolled Forward

Element identifier ts_name

Element type information

Table 347. 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:

- "Rollforward Timestamp" on page 323

Rollforward Type

Element identifier rf_type

Element type information

Table 348. 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.

Log Being Rolled Forward

Element identifier rf_log_num

Element type information

Table 349. 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.

Log Phase

Element identifier rf_status

Element type information

Table 350. 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:

- Table Space Identification
- Table Space Name
- Table Space Type
- Table Space Contents Type
- Table Space State
- Table Space Page Size
- Table Space Extent Size
- Table Space Prefetch Size
- Buffer Pool Currently Being Used
- Buffer Pool That Will Be Used at Next Startup
- Total Pages in Table Space
- Usable Pages in Table Space
- Used Pages in Table Space
- Free Pages in Table Space
- Pending Free Pages in Table Space
- Table Space High Water Mark
- Rebalancer Mode
- Rebalancer Start Time
- Rebalancer Restart Time
- Total Number of Extents to be Processed by the Rebalancer
- Number of Extents the Rebalancer has Processed
- Last Extent Moved by the Rebalancer
- Current Rebalancer Priority
- Number of Quiescers

- Table space quiescer activity data elements
- State Change Object Identification
- State Change Table Space Identification
- Minimum Recovery Time For Rollforward
- Number of Containers in Table Space
- Container status data elements
- Number of Ranges in the Table Space Map
- Table space range status data elements

Table Space Identification

Element identifier tablespace_id

Element type information

Table 351. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

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.

Table Space Name

Element identifier tablespace_name

Element type information

Table 352. 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 353. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Deadlocks	lock	-
Deadlocks	event_dlconn	-

Table 353. Event Monitoring Information (continued)

Event Type	Logical Data Grouping	Monitor Switch
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.

Related reference:

- "Lock Object Type Waited On" on page 304

Table Space Type

Element identifier tablespace_type

Element type information

Table 354. 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:

- For DMS: SQLM_TABLESPACE_TYP_DMS
- For SMS: SQLM_TABLESPACE_TYP_SMS

Table Space Contents Type

Element identifier tablespace_content_type

Element type information

Table 355. 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:

- any data: SQLM_TABLESPACE_CONTENT_ANY
- long data: SQLM_TABLESPACE_CONTENT_LONG
- system temporary data: SQLM_TABLESPACE_CONTENT_SYSTEMP
- user temporary data: SQLM_TABLESPACE_CONTENT_USRTEMP

Table Space State

Element identifier tablespace_state

Element type information

Table 356. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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. Following are the bit definitions listed in sqlutil.h:

- 0x0 Normal (see the definition SQLB_NORMAL in sqlutil.h)
- 0x1 Quiesced: SHARE
- 0x2 Quiesced: UPDATE
- 0x4 Quiesced: EXCLUSIVE
- 0x8 Load pending

- 0x10 Delete pending
- 0x20 Backup pending
- 0x40 Roll forward in progress
- 0x80 Roll forward pending
- 0x100 Restore pending
- 0x100 Recovery pending (not used)
- 0x200 Disable pending
- 0x400 Reorg in progress
- 0x800 Backup in progress
- 0x1000 Storage must be defined
- 0x2000 Restore in progress
- 0x4000 Offline and not accessible
- 0x8000 Drop pending
- 0x2000000 Storage may be defined
- 0x4000000 Storage Definition is in 'final' state
- 0x8000000 Storage Definition was changed prior to rollforward
- 0x10000000 DMS rebalancer is active
- 0x20000000 TBS deletion in progress
- 0x40000000 TBS creation in progress

Table Space Page Size

Element identifier tablespace_page_size

Element type information

Table 357. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

Page size used by a table space in bytes.

Table Space Extent Size

Element identifier tablespace_extent_size

Element type information

Table 358. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description
The extent size used by a table space.

Table Space Prefetch Size

Element identifier tablespace_prefetch_size

Element type information

Table 359. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description
The maximum number of pages the prefetcher gets from the disk at a time.

Buffer Pool Currently Being Used

Element identifier tablespace_cur_pool_id

Element type information

Table 360. 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.

Buffer Pool That Will Be Used at Next Startup

Element identifier tablespace_next_pool_id

Element type information

Table 361. 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

Total Pages in Table Space

Element identifier tablespace_total_pages

Element type information

Table 362. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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).

Usable Pages in Table Space

Element identifier tablespace_usable_pages

Element type information

Table 363. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Used Pages in Table Space

Element identifier tablespace_used_pages

Element type information

Table 364. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

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.

Free Pages in Table Space

Element identifier tablespace_free_pages

Element type information

Table 365. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Pending Free Pages in Table Space

Element identifier tablespace_pending_free_pages

Element type information

Table 366. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Table Space High Water Mark

Element identifier tablespace_page_top

Element type information

Table 367. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Rebalancer Mode

Element identifier tablespace_rebalancer_mode

Element type information

Table 368. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Rebalancer Start Time

Element identifier tablespace_rebalancer_start_time

Element type information

Table 369. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Rebalancer Restart Time

Element identifier tablespace_rebalancer_restart_time

Element type information

Table 370. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Total Number of Extents to be Processed by the Rebalancer

Element identifier tablespace_rebalancer_extents_remaining

Element type information

Table 371. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Number of Extents the Rebalancer has Processed

Element identifier tablespace_rebalancer_extents_processed

Element type information

Table 372. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Last Extent Moved by the Rebalancer

Element identifier tablespace_rebalancer_last_extent_moved

Element type information

Table 373. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Current Rebalancer Priority

Element identifier tablespace_rebalancer_priority

Element type information

Table 374. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	Basic

Description

The priority at which the rebalancer is running in the database.

Usage This is only applicable to a DMS table space.

Number of Quiescers

Element identifier tablespace_num_quiescers

Element type information

Table 375. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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 User Authorization Identification
- Quiescer Agent Identification

- Quiescer Table Space Identification
- Quiescer Object Identification
- Quiescer State

Quiescer User Authorization Identification:

Element identifier quiescer_auth_id

Element type information

Table 376. 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 Identification:

Element identifier quiescer_agent_id

Element type information

Table 377. 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 Table Space Identification:

Element identifier quiescer_ts_id

Element type information

Table 378. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace_quiescer	Basic

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 Object Identification:

Element identifier quiescer_obj_id

Element type information

Table 379. 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:

Element identifier quiescer_state

Element type information

Table 380. 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.

State Change Object Identification

Element identifier tablespace_state_change_object_id

Element type information

Table 381. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

State Change Table Space Identification

Element identifier tablespace_state_change_ts_id

Element type information

Table 382. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Minimum Recovery Time For Rollforward

Element identifier tablespace_min_recovery_time

Element type information

Table 383. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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.

Number of Containers in Table Space

Element identifier tablespace_num_containers

Element type information

Table 384. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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 Identification
- Container Name
- Container Type
- Total Pages in Container
- Usable Pages in Container
- Stripe Set
- Accessibility of Container

Container Identification:

Element identifier container_id

Element type information

Table 385. 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:

Element identifier container_name

Element type information

Table 386. 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:

Element identifier container_type

Element type information

Table 387. 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

Total Pages in Container:

Element identifier container_total_pages

Element type information

Table 388. 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.

Usable Pages in Container:

Element identifier container_usable_pages

Element type information

Table 389. 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.

Stripe Set:

Element identifier container_stripe_set

Element type information

Table 390. 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.

Accessibility of Container:

Element identifier container_accessible

Element type information

Table 391. 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.

Number of Ranges in the Table Space Map

Element identifier tablespace_num_ranges

Element type information

Table 392. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table Space	tablespace	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)
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):

- Stripe Set Number
- Range Number
- Maximum Page in Range
- Maximum Extent in Range
- Start Stripe
- End Stripe
- Range Adjustment
- Number of Containers in Range
- 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
  - Range Offset

Stripe Set Number:

Element identifier range_stripe_set_number

Element type information

Table 393. 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:

Element identifier range_number

Element type information

Table 394. 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.

Maximum Page in Range:

Element identifier range_max_page_number

Element type information

Table 395. 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.

Maximum Extent in Range:

Element identifier range_max_extent

Element type information

Table 396. 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.

Start Stripe:

Element identifier range_start_stripe

Element type information

Table 397. 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.

End Stripe:

Element identifier range_end_stripe

Element type information

Table 398. 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:

Element identifier range_adjustment

Element type information

Table 399. 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.

Number of Containers in Range:

Element identifier range_num_containers

Element type information

Table 400. 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:

Element identifier range_container_id

Element type information

Table 401. 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:

Element identifier range_offset

Element type information

Table 402. 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.

Table activity

Table activity monitor elements

The following elements provide information about the tables:

- Table Type
- Table Name
- Table Schema Name
- Rows Deleted

- Rows Inserted
- Rows Updated
- Rows Selected
- Rows Written
- Rows Read
- Accesses to Overflowed Records
- Internal Rows Deleted
- Internal Rows Updated
- Internal Rows Inserted
- Table File ID
- Page Reorganizations

Table Type

Element identifier table_type

Element type information

Table 403. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table

Table 404. 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. The table type will only be updated after the changes are committed (either explicitly or implicitly).
- 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.
- Reorganization table. A table created and used by the database manager while performing a reorganization of another table.

Related reference:

- "Table File ID" on page 359

Table Name

Element identifier table_name

Element type information

Table 405. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table
Application	appl	Lock
Lock	appl_lock_list	Lock
Lock	lock	Lock
Lock	lock_wait	Lock

Table 406. 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.

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. This element is blank for temporary tables, reorganization tables, and tables that were dropped. Table names are only provided for catalog and user tables. For

snapshot monitoring, this element is only valid when the "table" monitor group information is turned on.

Related reference:

- "Lock Object Type Waited On" on page 304
- "Lock Object Name" on page 305
- "Table Schema Name" on page 350

Table Schema Name

Element identifier table_schema

Element type information

Table 407. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table
Application	appl	Lock
Lock	appl_lock_list	Lock
Lock	lock	Lock
Lock	lock_wait	Lock

Table 408. 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. This element is blank for temporary tables, reorganization tables, and tables that were dropped. Schema names are provided only for catalog and user tables. For snapshot monitoring, this element is valid only when the "table" monitor group information is turned on.

Related reference:

- "Lock Object Type Waited On" on page 304
- "Lock Object Name" on page 305
- "Table Name" on page 349

Rows Deleted

Element identifier rows_deleted

Element type counter

Table 409. 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 410. 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:

• "Internal Rows Deleted" on page 357

Rows Inserted

Element identifier rows_inserted

Element type counter

Table 411. 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 412. 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

Element identifier rows_updated

Element type counter

Table 413. 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 414. 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.

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:

- "Internal Rows Updated" on page 358

Rows Selected

Element identifier rows_selected

Element type counter

Table 415. 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 416. 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:

```
average time = rows returned / 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 UDB version 7.2 or lower.

Related reference:

- "Select SQL Statements Executed" on page 376

Rows Written

Element identifier rows_written

Element type counter

Table 417. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
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 418. 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" on page 355
- "Internal Rows Deleted" on page 357
- "Internal Rows Updated" on page 358
- "Internal Rows Inserted" on page 359

Rows Read

Element identifier rows_read

Element type counter

Table 419. 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 420. 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" on page 354
- "Accesses to Overflowed Records" on page 356

Accesses to Overflowed Records

Element identifier overflow_accesses

Element type counter

Table 421. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table

For snapshot monitoring, this counter can be reset.

Table 422. 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

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" on page 354
- "Rows Read" on page 355

Internal Rows Deleted

Element identifier int_rows_deleted

Element type counter

Table 423. 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 424. 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" on page 351

Internal Rows Updated

Element identifier int_rows_updated

Element type counter

Table 425. 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 426. 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" on page 352

Internal Rows Inserted

Element identifier int_rows_inserted

Element type counter

Table 427. 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 428. Event Monitoring Information

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

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" on page 352

Table File ID

Element identifier table_file_id

Element type information

Table 429. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Lock
Table	table	Table
Lock	appl_lock_list	Lock
Lock	lock	Lock

Table 430. 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 Type" on page 348
- "Table Name" on page 349
- "Table Schema Name" on page 350

Page Reorganizations

Element identifier page_reorgs

Element type counter

Table 431. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table	Table

For snapshot monitoring, this counter can be reset.

Table 432. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Tables	event_table	-

Description

The number of page reorganizations executed for a table.

Usage 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 and so 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. Or if the page

does not have enough space for the new larger row, an overflow record is created being 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" on page 352
- "Rows Updated" on page 352

Table reorganization

Table reorganization monitor elements

The following elements provide information about table reorganization:

- Table Reorganize Attribute Flag
- Table Reorganize Status
- Table Reorganize Phase
- Table Reorganize Phase Start Time
- Maximum Table Reorganize Phase
- Percentage Complete of Table Reorganize
- Total Number of Pages in Object
- Table Reorganize Completion Flag
- Table Reorganize Start Time
- Table Reorganize End Time
- Reorg Index
- Reorg Table Space

Table Reorganize Attributes

Element identifier reorg_type

Element type information

Table 433. 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_NOTRUNCATE_ONLINE

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.

Table Reorganize Status

Element identifier reorg_status

Element type information

Table 434. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The status of an in-place (online) table reorganization. This is not applicable to classic (offline) table reorganizations.

Usage An in-place table 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

Table Reorganize Phase

Element identifier reorg_phase

Element type information

Table 435. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

Table reorganize phase. This applies to classic (offline) table reorganizations only).

Usage 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

Table Reorganize Phase Start Time

Element identifier reorg_phase_start

Element type timestamp

Table 436. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The start time of a phase of table reorganization.

Maximum Table Reorganize Phase

Element identifier reorg_max_phase

Element type information

Table 437. 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.

Table Reorganize Progress

Element identifier reorg_current_counter

Element type information

Table 438. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

A unit of progress that indicates the amount of table 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:

```
table reorg progress = reorg_current_counter / reorg_max_counter * 100
```

Total Amount of Table Reorganization

Element identifier reorg_max_counter

Element type information

Table 439. 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 table reorganization. This value can be used with reorg_current_counter, which represents the amount of work completed, to determine the progress of a table reorganization.

Table Reorganization Completion Flag

Element identifier reorg_completion

Element type information

Table 440. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

Table reorganize success indicator.

Usage This element will have a value of 0 if a table reorganize operation is successful. If a table 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. See the administration notification log for further diagnostic information.

Table Reorganize Start Time

Element identifier reorg_start

Element type timestamp

Table 441. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The start time of a table reorganization.

Table Reorganize End Time

Element identifier reorg_end

Element type timestamp

Table 442. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description

The end time of a table reorganization.

Index Used to Reorganize the Table

Element identifier reorg_index_id

Element type information

Table 443. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Table	table_reorg	Basic

Description
The index being used to reorganize the table.

Table Space Where Table is Reorganized

Element identifier reorg_tbspc_id

Element type information

Table 444. 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.

SQL cursors

SQL cursors monitor elements

The following elements provide information about the SQL cursors:

- Open Remote Cursors
- Open Remote Cursors with Blocking
- Rejected Block Cursor Requests
- Accepted Block Cursor Requests
- Open Local Cursors
- Open Local Cursors with Blocking

Open Remote Cursors

Element identifier open_rem_curs

Element type gauge

Table 445. 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 Remote Cursors with Blocking" on page 367
- "Open Local Cursors" on page 369

Open Remote Cursors with Blocking

Element identifier open_rem_curs_blk

Element type gauge

Table 446. 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 Remote Cursors" on page 366
- "Rejected Block Cursor Requests" on page 368
- "Accepted Block Cursor Requests" on page 368
- "Open Local Cursors" on page 369
- "Open Local Cursors with Blocking" on page 370

Rejected Block Cursor Requests

Element identifier rej_curs_blk

Element type counter

Table 447. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 448. 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:

- "Open Remote Cursors" on page 366
- "Open Remote Cursors with Blocking" on page 367
- "Accepted Block Cursor Requests" on page 368
- "Open Local Cursors" on page 369
- "Open Local Cursors with Blocking" on page 370

Accepted Block Cursor Requests

Element identifier acc_curs_blk

Element type counter

Table 449. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Basic

Table 450. 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 and/or rejected.

See *rej_curs_blk* for suggestions on how to use this information to tune your configuration parameters.

Related reference:

- "Open Remote Cursors" on page 366
- "Open Remote Cursors with Blocking" on page 367
- "Rejected Block Cursor Requests" on page 368
- "Open Local Cursors" on page 369
- "Open Local Cursors with Blocking" on page 370

Open Local Cursors

Element identifier open_loc_curs

Element type gauge

Table 451. 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 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 Remote Cursors" on page 366
- "Open Remote Cursors with Blocking" on page 367
- "Rejected Block Cursor Requests" on page 368

- "Accepted Block Cursor Requests" on page 368
- "Open Local Cursors with Blocking" on page 370

Open Local Cursors with Blocking

Element identifier open_loc_curs_blk

Element type gauge

Table 452. 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 Remote Cursors" on page 366
- "Open Remote Cursors with Blocking" on page 367
- "Rejected Block Cursor Requests" on page 368
- "Accepted Block Cursor Requests" on page 368
- "Open Local Cursors" on page 369

SQL statement activity

SQL statement activity monitor elements

The following elements provide information about SQL statement activity:

- Static SQL Statements Attempted

- Dynamic SQL Statements Attempted
- Failed Statement Operations
- Commit Statements Attempted
- Rollback Statements Attempted
- Select SQL Statements Executed
- Update/Insert/Delete SQL Statements Executed
- Data Definition Language (DDL) SQL Statements
- Internal Automatic Rebinds
- Internal Commits
- Internal Rollbacks
- Internal Rollbacks Due To Deadlock
- SQL Requests Since Last Commit
- Statement Node
- Binds/Precompiles Attempted

Static SQL Statements Attempted

Element identifier static_sql_stmts

Element type counter

Table 453. 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 454. 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:

```
  dynamic_sql_stmts
+ static_sql_stmts
- failed_sql_stmts
= throughput during monitoring period
```

Related reference:

- "Failed Statement Operations" on page 372

Dynamic SQL Statements Attempted

Element identifier dynamic_sql_stmts

Element type counter

Table 455. 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 456. 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 Statement Operations" on page 372

Failed Statement Operations

Element identifier failed_sql_stmts

Element type counter

Table 457. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic
Database	dbase_remote	Basic
Application	appl	Basic

Table 457. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl_remote	Basic
DCS Database	dcs_dbase	Basic
DCS Application	dcs_appl	Basic

For snapshot monitoring, this counter can be reset.

Table 458. 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:

- "Static SQL Statements Attempted" on page 371
- "Dynamic SQL Statements Attempted" on page 372

Commit Statements Attempted

Element identifier commit_sql_stmts

Element type counter

Table 459. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase	Basic

Table 459. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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 460. 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:

- 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:

- "Rollback Statements Attempted" on page 375
- "Internal Commits" on page 380
- "Internal Rollbacks" on page 381
- "Internal Rollbacks Due To Deadlock" on page 382

Rollback Statements Attempted

Element identifier rollback_sql_stmts

Element type counter

Table 461. 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 462. 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:

- "Commit Statements Attempted" on page 373
- "Internal Commits" on page 380
- "Internal Rollbacks" on page 381
- "Internal Rollbacks Due To Deadlock" on page 382
- "Statement Type" on page 385

Select SQL Statements Executed

Element identifier select_sql_stmts

Element type counter

Table 463. 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 464. 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:

```
  select_sql_stmts
/ ( static_sql_stmts
  + dynamic_sql_stmts )
```

This information can be useful for analyzing application activity and throughput.

Related reference:

- "Static SQL Statements Attempted" on page 371
- "Dynamic SQL Statements Attempted" on page 372

Update/Insert/Delete SQL Statements Executed

Element identifier uid_sql_stmts

Element type counter

Table 465. 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 466. 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:

```
  uid_sql_stmts
/ (static_sql_stmts + dynamic_sql_stmts )
```

This information can be useful for analyzing application activity and throughput.

Related reference:

- "Static SQL Statements Attempted" on page 371
- "Dynamic SQL Statements Attempted" on page 372

Data Definition Language (DDL) SQL Statements

Element identifier ddl_sql_stmts

Element type counter

Table 467. 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 468. 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:

```
ddl_sql_stmts / total number of statements
```

This information can be useful for analyzing application activity and throughput. DDL statements can also impact:

- the catalog cache, by invalidating table descriptor information and authorization information that are stored there and causing additional system 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.

Internal Automatic Rebinds

Element identifier int_auto_rebinds

Element type counter

Table 469. 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 470. 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 an package has been invalidated. The rebind is performed the first time that the database manager needs to execute an SQL statement from the package. For example, packages are invalidated when you:

- Drop an object, such as a table, view, or index, on which the plan is dependent
- Add or drop a foreign key
- Revoke object privileges on which the plan is dependent.

You can use this element to determine the level of database activity at the application or database level. Since int_auto_rebinds can have a significant impact on performance, they should be minimized where possible.

You can also use this element to determine the percentage of rebind activity using the following formula:

```
int_auto_rebinds / total number of statements
```

This information can be useful for analyzing application activity and throughput.

Related reference:

• "Binds/Precompiles Attempted" on page 384

Internal Commits

Element identifier int_commits

Element type counter

Table 471. 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 472. Event Monitoring Information

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

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:

```
   commit_sql_stmts
 + int_commits
 + rollback_sql_stmts
 + int_rollbacks
```

Note: The units of work calculated will only include those since the later of:

- The connection to the database (for database-level information, this is the time of the first connection)
- The last reset of the database monitor counters.

This calculation can be done at the application or the database level.

Related reference:

- "Commit Statements Attempted" on page 373
- "Rollback Statements Attempted" on page 375
- "Internal Rollbacks" on page 381

Internal Rollbacks

Element identifier int_rollbacks

Element type counter

Table 473. 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 474. 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:

```
  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 Statements Attempted" on page 373
- "Rollback Statements Attempted" on page 375
- "Internal Commits" on page 380
- "Internal Rollbacks Due To Deadlock" on page 382

Internal Rollbacks Due To Deadlock

Element identifier int_deadlock_rollbacks

Element type counter

Table 475. 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 476. 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_rollbackss lower the throughput of the database.

This value is included in the value given by int_rollbacks.

Related reference:

- "Deadlocks Detected" on page 298
- "Rollback Statements Attempted" on page 375
- "Internal Rollbacks" on page 381

SQL Requests Since Last Commit

Element identifier sql_reqs_since_commit

Element type information

Table 477. 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.

Statement Node

Element identifier stmt_node_number

Element type information

Table 478. 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 Attempted

Element identifier binds_precompiles

Element type counter

Table 479. 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 480. 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:

- "Internal Automatic Rebinds" on page 379

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:

- Statement Type
- Statement Operation
- Package Name
- Package Version
- Package Consistency Token
- Section Number
- Cursor Name
- Application Creator

- Statement Operation Start Timestamp
- Statement Operation Stop Timestamp
- Event Stop Time
- Event Start Time
- Most Recent Statement Elapsed Time
- SQL Dynamic Statement Text
- Statement Sorts
- Number of Successful Fetches
- SQL Communications Area (SQLCA)
- Query Number of Rows Estimate
- Query Cost Estimate

Statement Type

Element identifier stmt_type

Element type information

Table 481. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Table 482. Event Monitoring Information

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

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.

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:

- "Package Name" on page 387
- "Section Number" on page 389
- "Application Creator" on page 391
- "SQL Dynamic Statement Text" on page 394
- "Package Version" on page 388

Statement Operation

Element identifier stmt_operation (snapshot monitoring)
operation (event monitoring)

Element type information

Table 483. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 484. 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:
- "Statement Type" on page 385
- "Package Name" on page 387
- "Section Number" on page 389
- "Application Creator" on page 391
- "SQL Dynamic Statement Text" on page 394
- "Number of Successful Fetches" on page 396
- "Package Version" on page 388

Package Name

Element identifier package_name

Element type information

Table 485. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 486. 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:

- "Section Number" on page 389
- "Application Creator" on page 391
- "SQL Dynamic Statement Text" on page 394
- "Package Version" on page 388

Package Consistency Token

Element identifier consistency_token

Element type information

Table 487. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Table 488. 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

Element identifier package_version_id

Element type information

Table 489. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Table 490. 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:

- "Package Name" on page 387
- "Section Number" on page 389
- "Application Creator" on page 391
- "SQL Dynamic Statement Text" on page 394

Section Number

Element identifier section_number

Element type information

Table 491. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 492. 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
             PKGSCHEMA = 'creator'      AND
             VERSION = 'package_version_id' AND
             SECTNO    = section_number
       ORDER BY SEQNO
```

Note: Exercise caution in obtaining static statement text, because this query against the system catalog table could cause lock contentions. Whenever possible, only use this query when there is little other activity against the database.

Related reference:

- "Package Name" on page 387
- "Application Creator" on page 391
- "SQL Dynamic Statement Text" on page 394
- "Package Version" on page 388

Cursor Name

Element identifier cursor_name

Element type information

Table 493. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement

Table 494. 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:

- "Statement Type" on page 385
- "SQL Dynamic Statement Text" on page 394
- "Number of Successful Fetches" on page 396

Application Creator

Element identifier creator

Element type information

Table 495. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 496. 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 You may 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.

Related reference:

- "Package Name" on page 387
- "Section Number" on page 389
- "Package Version" on page 388

Statement Operation Start Timestamp

Element identifier stmt_start

Element type timestamp

Table 497. 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:

- "Statement Operation" on page 386
- "Statement Operation Stop Timestamp" on page 392

Statement Operation Stop Timestamp

Element identifier stmt_stop

Element type Timestamp

Table 498. 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:

- "Statement Operation" on page 386
- "Statement Operation Start Timestamp" on page 391
- "Event Stop Time" on page 392

Event Stop Time

Element identifier stop_time

Element type timestamp

Table 499. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-

Description
The date and time when the statement stopped executing.

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.

Related reference:

- "Previous Transaction Stop Time" on page 196
- "Statement Operation Stop Timestamp" on page 392

Event Start Time

Element identifier start_time

Element type timestamp

Table 500. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Database	event_start	-
Transactions	event_xact	-
Statements	event_stmt	-
Deadlocks	event_deadlock	-
Deadlocks	event_dlconn	-
Deadlocks with Details	event_detailed_dlconn	-

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.

Related reference:

- "Previous Transaction Stop Time" on page 196
- "Statement Operation" on page 386

Most Recent Statement Elapsed Time

Element identifier stmt_elapsed_time

Element type time

Table 501. 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:

- "Communication Errors" on page 461
- "Communication Error Time" on page 462

SQL Dynamic Statement Text

Element identifier stmt_text

Element type information

Table 502. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
Dynamic SQL	dynsql	Basic
DCS Statement	dcs_stmt	Statement

Table 503. 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 snapshot 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, it is returned in the statement event record for all dynamic statements.

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:

- "Statement Operation" on page 386
- "Package Name" on page 387
- "Section Number" on page 389
- "Cursor Name" on page 390
- "Application Creator" on page 391
- "Input Database Alias" on page 419
- "Package Version" on page 388

Statement Sorts

Element identifier stmt_sorts

Element type counter

Table 504. 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" on page 221

Number of Successful Fetches

Element identifier fetch_count

Element type counter

Table 505. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 506. 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:

- "Statement Type" on page 385
- "Statement Operation" on page 386
- "Cursor Name" on page 390
- "Statement Operation Start Timestamp" on page 391
- "Statement Operation Stop Timestamp" on page 392

SQL Communications Area (SQLCA)

Element identifier sqlca

Element type information

Table 507. 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:

- "Statement Operation" on page 386

Query Number of Rows Estimate

Element identifier query_card_estimate

Element type information

Table 508. 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 Universal Database, 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" on page 398

Query Cost Estimate

Element identifier query_cost_estimate

Element type information

Table 509. 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.

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.

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:

- Subsection Number
- Subsection Node Number
- Subsection Status
- Subsection Execution Elapsed Time
- Number of Agents Working on a Subsection
- Waiting for Any Node to Send on a Tablequeue
- Waited for Node on a Tablequeue
- Total Number of Tablequeue Buffers Overflowed
- Current Number of Tablequeue Buffers Overflowed
- Number of Rows Read from Tablequeues
- Number of Rows Written to Tablequeues
- Maximum Number of Tablequeue Buffers Overflows
- Waited on Node on a Tablequeue

Subsection Number

Element identifier ss_number

Element type information

Table 510. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 511. 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.

Subsection Node Number

Element identifier ss_node_number

Element type information

Table 512. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 513. 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.

Subsection Status

Element identifier ss_status

Element type information

Table 514. 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:
- "Waiting for Any Node to Send on a Tablequeue" on page 401
- "Waited for Node on a Tablequeue" on page 402

Subsection Execution Elapsed Time

Element identifier ss_exec_time

Element type counter

Table 515. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 516. 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.

Waiting for Any Node to Send on a Tablequeue

Element identifier tq_wait_for_any

Element type information

Table 517. 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 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:

- "Subsection Status" on page 400
- "Waited for Node on a Tablequeue" on page 402

Waited for Node on a Tablequeue

Element identifier tq_node_waited_for

Element type information

Table 518. 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:

- "Subsection Status" on page 400
- "Waiting for Any Node to Send on a Tablequeue" on page 401

Total Number of Tablequeue Buffers Overflowed

Element identifier tq_tot_send_spills

Element type counter

Table 519. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 520. 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:

- "Subsection Status" on page 400
- "Current Number of Tablequeue Buffers Overflowed" on page 403

Current Number of Tablequeue Buffers Overflowed

Element identifier tq_cur_send_spills

Element type gauge

Table 521. 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 your 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:

- "Subsection Status" on page 400
- "Total Number of Tablequeue Buffers Overflowed" on page 402

Number of Rows Read from Tablequeues

Element identifier tq_rows_read

Element type counter

Table 522. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 523. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_subsection	-

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.

Number of Rows Written to Tablequeues

Element identifier tq_rows_written

Element type counter

Table 524. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 525. 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.

Maximum Number of Tablequeue Buffers Overflows

Element identifier tq_max_send_spills

Element type water mark

Table 526. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Statement

Table 527. 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:

- "Total Number of Tablequeue Buffers Overflowed" on page 402
- "Current Number of Tablequeue Buffers Overflowed" on page 403

Waited on Node on a Tablequeue

Element identifier tq_id_waiting_on

Element type information

Table 528. 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:

- "Subsection Status" on page 400
- "Waited for Node on a Tablequeue" on page 402

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.

- Statement Executions
- Statement Compilations
- Statement Worst Preparation Time
- Statement Best Preparation Time
- Elapsed Statement Execution Time

Statement Executions

Element identifier num_executions

Element type counter

Table 529. 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.

Related reference:

- "Statement Compilations" on page 407

Statement Compilations

Element identifier num_compilations

Element type counter

Table 530. 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 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:

- "Statement Executions" on page 406

Statement Worst Preparation Time

Element identifier prep_time_worst

Element type water mark

Table 531. 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:

- "Statement Best Preparation Time" on page 408

Statement Best Preparation Time

Element identifier prep_time_best

Element type water mark

Table 532. 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:

- "Statement Worst Preparation Time" on page 407

Elapsed Statement Execution Time

Element identifier total_exec_time

Element type time

Table 533. 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:

- "Statement Executions" on page 406
- "Statement Compilations" on page 407
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

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:

- Number of Agents Working on a Statement
- Number of Agents Created
- Degree of Parallelism

Number of Agents Working on a Statement

Element identifier num_agents

Element type gauge

Table 534. 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.

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:

- "Number of Agents Created" on page 409
- "Degree of Parallelism" on page 410

Number of Agents Created

Element identifier agents_top

Element type water mark

Table 535. 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:

- "Number of Agents Working on a Statement" on page 409
- "Degree of Parallelism" on page 410

Degree of Parallelism

Element identifier degree_parallelism

Element type information

Table 536. 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:

- "Number of Agents Working on a Statement" on page 409
- "Number of Agents Created" on page 409

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.

- User CPU Time used by Agent
- System CPU Time used by Agent
- User CPU Time used by Statement
- System CPU Time used by Statement
- User CPU Time
- System CPU Time
- User CPU Time used by Subsection
- System CPU Time used by Subsection
- Total System CPU for a Statement

- Total User CPU for a Statement

User CPU Time used by Agent

Element identifier agent_usr_cpu_time

Element type time

Table 537. 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 fenced 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 returned as 0.

Related reference:

- "System CPU Time used by Agent" on page 411
- "User CPU Time used by Statement" on page 412
- "System CPU Time used by Statement" on page 413
- "User CPU Time" on page 414
- "System CPU Time" on page 415
- "User CPU Time used by Subsection" on page 416
- "System CPU Time used by Subsection" on page 417
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

System CPU Time used by Agent

Element identifier agent_sys_cpu_time

Element type time

Table 538. 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.

It includes CPU time for both SQL and non-SQL statements, as well as CPU time for any fenced 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:

- "User CPU Time used by Agent" on page 411
- "User CPU Time used by Statement" on page 412
- "System CPU Time used by Statement" on page 413
- "User CPU Time" on page 414
- "System CPU Time" on page 415
- "User CPU Time used by Subsection" on page 416
- "System CPU Time used by Subsection" on page 417
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

User CPU Time used by Statement

Element identifier stmt_usr_cpu_time

Element type time

Table 539. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	appl	Statement, Timestamp

Table 539. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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 fenced 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:

- "User CPU Time used by Agent" on page 411
- "System CPU Time used by Agent" on page 411
- "System CPU Time used by Statement" on page 413
- "User CPU Time" on page 414
- "System CPU Time" on page 415
- "User CPU Time used by Subsection" on page 416
- "System CPU Time used by Subsection" on page 417
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

System CPU Time used by Statement

Element identifier stmt_sys_cpu_time

Element type time

Table 540. 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 fenced 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:

- "User CPU Time used by Agent" on page 411
- "System CPU Time used by Agent" on page 411
- "User CPU Time used by Statement" on page 412
- "User CPU Time" on page 414
- "System CPU Time" on page 415
- "User CPU Time used by Subsection" on page 416
- "System CPU Time used by Subsection" on page 417
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

User CPU Time

Element identifier user_cpu_time

Element type time

Table 541. 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:

- "User CPU Time used by Agent" on page 411
- "System CPU Time used by Agent" on page 411
- "User CPU Time used by Statement" on page 412
- "System CPU Time used by Statement" on page 413
- "System CPU Time" on page 415
- "User CPU Time used by Subsection" on page 416
- "System CPU Time used by Subsection" on page 417
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

System CPU Time

Element identifier system_cpu_time

Element type time

Table 542. 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 help 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:

- "User CPU Time used by Agent" on page 411
- "System CPU Time used by Agent" on page 411
- "User CPU Time used by Statement" on page 412
- "System CPU Time used by Statement" on page 413
- "User CPU Time" on page 414
- "User CPU Time used by Subsection" on page 416
- "System CPU Time used by Subsection" on page 417
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

User CPU Time used by Subsection

Element identifier ss_usr_cpu_time

Element type time

Table 543. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Timestamp

Table 544. 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:

- "User CPU Time used by Agent" on page 411
- "System CPU Time used by Agent" on page 411
- "User CPU Time used by Statement" on page 412
- "System CPU Time used by Statement" on page 413
- "User CPU Time" on page 414
- "System CPU Time" on page 415

- "System CPU Time used by Subsection" on page 417
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

System CPU Time used by Subsection

Element identifier ss_sys_cpu_time

Element type time

Table 545. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	subsection	Timestamp

Table 546. 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:

- "User CPU Time used by Agent" on page 411
- "System CPU Time used by Agent" on page 411
- "User CPU Time used by Statement" on page 412
- "System CPU Time used by Statement" on page 413
- "User CPU Time" on page 414
- "User CPU Time used by Subsection" on page 416
- "Total System CPU for a Statement" on page 417
- "Total User CPU for a Statement" on page 418

Total System CPU for a Statement

Element identifier total_sys_cpu_time

Element type time

Table 547. 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:

- "Elapsed Statement Execution Time" on page 408
- "Total User CPU for a Statement" on page 418

Total User CPU for a Statement

Element identifier total_usr_cpu_time

Element type time

Table 548. 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:

- "Elapsed Statement Execution Time" on page 408
- "Total System CPU for a Statement" on page 417

Snapshot monitoring

Snapshot monitoring monitor elements

The following elements provide information about monitoring applications. They are returned as output for every snapshot:

- Last Reset Timestamp
- Input Database Alias
- Snapshot Time
- Number of Nodes in Partition

Last Reset Timestamp

Element identifier last_reset

Element type timestamp

Table 549. 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.

The database manager counters will only be reset if you reset all active databases.

Related reference:

- "Input Database Alias" on page 419

Input Database Alias

Element identifier input_db_alias

Element type information

Table 550. 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:

- "Database Alias Used by Application" on page 181
- "Last Reset Timestamp" on page 419

Snapshot Time

Element identifier time_stamp

Element type timestamp

Table 551. 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.

Number of Nodes in Partition

Element identifier num_nodes_in_db2_instance

Element type information

Table 552. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic

Table 553. 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:

- Number of Event Monitor Overflows
- Time of First Event Overflow
- Time of Last Event Overflow
- Byte Order of Event Data
- Version of Monitor Data
- Event Monitor Name
- Partial Record
- Event Time
- Number of Event Monitor Flushes
- Number of Event Monitor Activations
- Request Identifier for SQL Statement
- Control Table Message
- Timestamp Control Table Message
- Partition Number

Number of Event Monitor Overflows

Element identifier count

Element type counter

Table 554. 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.

The event monitor sends one overflow record for a set of consecutive overflows.

Time of First Event Overflow

Element identifier first_overflow_time

Element type timestamp

Table 555. 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_over_flow time* to calculate the elapsed time for which the overflow record was generated.

Related reference:

- "Number of Event Monitor Overflows" on page 421

Time of Last Event Overflow

Element identifier last_overflow_time

Element type timestamp

Table 556. 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:

- "Number of Event Monitor Overflows" on page 421

Byte Order of Event Data

Element identifier byte_order

Element type information

Table 557. 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 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 of Monitor Data

Element identifier version

Element type information

Table 558. 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

Element identifier event_monitor_name

Element type information

Table 559. 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

Element identifier partial_record

Element type information

Table 560. 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

Element identifier event_time

Element type information

Table 561. 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.

Number of Event Monitor Flushes

Element identifier evmon_flushes

Element type information

Table 562. 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.

Number of Event Monitor Activations

Element identifier evmon_activates

Element type counter

Table 563. 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.

When started, an 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.

A DEADLOCKS WITH DETAILS event monitor is created for each newly created database. This event monitor is created with the AUTOSTART option.

Request Identifier for SQL Statement

Element identifier sql_req_id

Element type information

Table 564. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
Statements	event_stmt	-

Description
The request identifier for an SQL statement.

Usage This identifier increments with each successive SQL request 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.

Control Table Message

Element identifier message

Element type information

Table 565. 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.

Timestamp Control Table Message

Element identifier message_time

Element type timestamp

Table 566. 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

Element identifier partition_number

Element type information

Table 567. Event Monitoring Information

Event Type	Logical Data Grouping	Monitor Switch
-	-	-

Description

This element is only used in the CONTROL table 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.

DB2 Connect

DB2 Connect monitor elements

The following elements provide DB2 Connection information at the database, application, transaction, and statement levels:

- DCS Database Name
- Host Database Name
- Database Alias at the Gateway
- DB2 Connect Gateway First Connect Initiated
- Maximum Number of Concurrent Connections to Host Database
- Total Number of Attempted Connections for DB2 Connect
- Current Number of Connections for DB2 Connect
- Number of Connections Waiting for the Host to Reply
- Number of Connections Waiting for the Client to Send Request
- Elapsed Time Spent on DB2 Connect Gateway Processing
- Number of SQL Statements Attempted
- Number of Open Cursors
- DCS Application Status
- DCS Application Agents
- Host Coded Character Set ID
- Outbound Communication Protocol
- Outbound Communication Address
- Inbound Communication Address
- Inbound Number of Bytes Received
- Outbound Number of Bytes Sent

- Outbound Number of Bytes Received
- Inbound Number of Bytes Sent
- Maximum Outbound Number of Bytes Sent
- Maximum Outbound Number of Bytes Received
- Minimum Outbound Number of Bytes Sent
- Minimum Outbound Number of Bytes Received
- Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes
- Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes
- Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes
- Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes
- Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes
- Number of Statements with Outbound Bytes Received Between 257 and 512 Bytes
- Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes
- Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes
- Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes
- Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes
- Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes
- Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes
- Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes
- Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes
- Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes
- Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes
- Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes

- Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes
- Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes
- Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes
- Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes
- Number of Statements with Outbound Bytes Received Greater than 64000 Bytes
- Number of Statements with Network Time of up to 2 ms
- Number of Statements with Network Time between 2 and 4 ms
- Number of Statements with Network Time between 4 and 8 ms
- Number of Statements with Network Time between 8 and 16 ms
- Number of Statements with Network Time between 16 and 32 ms
- Number of Statements with Network Time greater than 32 ms
- Maximum Network Time for Statement
- Minimum Network Time for Statement
- Transaction ID
- Elapsed Execution Time
- Host Response Time
- Number of Transmissions
- Most Recent Response Time for Connect
- Most Recent Connection Elapsed Time
- Communication Errors
- Communication Error Time
- Blocking Cursor
- Outbound Blocking Cursor
- Statement Execution Elapsed Time
- Transaction processor monitoring data elements

DCS Database Name

Element identifier dcs_db_name

Element type information

Table 568. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_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 Database Name" on page 431
- "Database Alias at the Gateway" on page 431

Host Database Name

Element identifier host_db_name

Element type information

Table 569. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Basic
DCS Application	dcs_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:
- "DCS Database Name" on page 430
- "Database Alias at the Gateway" on page 431

Database Alias at the Gateway

Element identifier gw_db_alias

Element type information

Table 570. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_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 Database Name" on page 430
- "Host Database Name" on page 431

DB2 Connect Gateway First Connect Initiated

Element identifier gw_con_time

Element type timestamp

Table 571. 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.

Maximum Number of Concurrent Connections to Host Database

Element identifier gw_connections_top

Element type water mark

Table 572. 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:

- "Total Number of Attempted Connections for DB2 Connect" on page 432
- "Current Number of Connections for DB2 Connect" on page 433

Total Number of Attempted Connections for DB2 Connect

Element identifier gw_total_cons

Element type water mark

Table 573. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database Manager	db2	Basic
DCS Database	dcs_dbase	Basic

For snapshot monitoring, this counter can be reset.

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:

- "Maximum Number of Concurrent Connections to Host Database" on page 432
- "Current Number of Connections for DB2 Connect" on page 433

Current Number of Connections for DB2 Connect

Element identifier gw_cur_cons

Element type gauge

Table 574. 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:

- "Maximum Number of Concurrent Connections to Host Database" on page 432
- "Total Number of Attempted Connections for DB2 Connect" on page 432

Number of Connections Waiting for the Host to Reply

Element identifier gw_cons_wait_host

Element type gauge

Table 575. 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:

- "Current Number of Connections for DB2 Connect" on page 433
- "Number of Connections Waiting for the Client to Send Request" on page 434

Number of Connections Waiting for the Client to Send Request

Element identifier gw_cons_wait_client

Element type gauge

Table 576. 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:

- "Current Number of Connections for DB2 Connect" on page 433
- "Number of Connections Waiting for the Host to Reply" on page 433

Elapsed Time Spent on DB2 Connect Gateway Processing

Element identifier gw_exec_time

Element type time

Table 577. 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.

Usage Use this element to determine what portion of the overall processing time is due to DB2 Connect gateway processing.

Number of SQL Statements Attempted

Element identifier sql_stmts

Element type counter

Table 578. 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
The number of SQL statements that have been attempted since the latter of: application start up, database activation, or last reset.

For the data transmission level: Number of SQL statements that have been attempted against this DCS database or in this DCS application since either the database was activated, or the connection to it established by the application, or RESET MONITOR was issued against the database, and which used this number of data transmissions between the DB2 Connect gateway and the host during statement processing.

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 etc. 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.

Related reference:

- "Snapshot Time" on page 420

Number of Open Cursors

Element identifier open_cursors

Element type gauge

Table 579. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_appl	Basic

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 `RQRIOBLK`. If *deferred_prepare* is enabled, then two buffers will be allocated.

DCS Application Status

Element identifier dcs_appl_status

Element type information

Table 580. 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 Coded Character Set ID" on page 438
- "Outbound Communication Protocol" on page 438
- "Outbound Communication Address" on page 439
- "Inbound Communication Address" on page 439

DCS Application Agents

Element identifier agent_status

Element type information

Table 581. 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 Application Status" on page 436

Host Coded Character Set ID

Element identifier host_ccsid

Element type information

Table 582. 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 Application Status" on page 436
- "Outbound Communication Protocol" on page 438
- "Outbound Communication Address" on page 439
- "Inbound Communication Address" on page 439

Outbound Communication Protocol

Element identifier outbound_comm_protocol

Element type information

Table 583. 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 Application Status" on page 436
- "Host Coded Character Set ID" on page 438
- "Outbound Communication Address" on page 439
- "Inbound Communication Address" on page 439

Outbound Communication Address

Element identifier outbound_comm_address

Element type information

Table 584. 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 Application Status" on page 436
- "Host Coded Character Set ID" on page 438
- "Outbound Communication Protocol" on page 438
- "Inbound Communication Address" on page 439

Inbound Communication Address

Element identifier inbound_comm_address

Element type information

Table 585. 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 Application Status" on page 436
- "Host Coded Character Set ID" on page 438
- "Outbound Communication Protocol" on page 438

- "Outbound Communication Address" on page 439

Inbound Number of Bytes Received

Element identifier inbound_bytes_received

Element type counter

Table 586. 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 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 Number of Bytes Sent" on page 440
- "Outbound Number of Bytes Received" on page 441
- "Inbound Number of Bytes Sent" on page 442

Outbound Number of Bytes Sent

Element identifier outbound_bytes_sent

Element type counter

Table 587. 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 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 Number of Bytes Received" on page 440
- "Outbound Number of Bytes Received" on page 441
- "Inbound Number of Bytes Sent" on page 442

Outbound Number of Bytes Received

Element identifier outbound_bytes_received

Element type counter

Table 588. 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 Number of Bytes Received" on page 440
- "Outbound Number of Bytes Sent" on page 440

- "Inbound Number of Bytes Sent" on page 442

Inbound Number of Bytes Sent

Element identifier inbound_bytes_sent

Element type counter

Table 589. 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 Number of Bytes Received" on page 440
- "Outbound Number of Bytes Sent" on page 440
- "Outbound Number of Bytes Received" on page 441

Maximum Outbound Number of Bytes Sent

Element identifier outbound_bytes_sent_top

Element type water mark

Table 590. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description

Maximum number of bytes sent per statement by the DB2 Connect gateway to the host during the processing of all the statements 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.

Maximum Outbound Number of Bytes Received

Element identifier outbound_bytes_received_top

Element type water mark

Table 591. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description

Maximum number of bytes received per statement by the DB2 Connect gateway from the host during the processing of all the statements 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.

Minimum Outbound Number of Bytes Sent

Element identifier outbound_bytes_sent_bottom

Element type water mark

Table 592. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description

The lowest number of bytes sent per statement by the DB2 Connect gateway to the host during the processing of all the statements 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.

Minimum Outbound Number of Bytes Received

Element identifier outbound_bytes_received_bottom

Element type water mark

Table 593. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Data Transmission	stmt_transmissions	Statement

Description
The lowest number of bytes received per statement by the DB2 Connect gateway from the host during the processing of all the statements 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.

Number of Statements with Outbound Bytes Sent Between 1 and 128 Bytes

Element identifier max_data_sent_128

Element type counter

Table 594. 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 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.

Number of Statements with Outbound Bytes Received Between 1 and 128 Bytes

Element identifier max_data_received_128

Element type counter

Table 595. 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 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.

Number of Statements with Outbound Bytes Sent Between 129 and 256 Bytes

Element identifier max_data_sent_256

Element type counter

Table 596. 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 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.

Number of Statements with Outbound Bytes Received Between 129 and 256 Bytes

Element identifier max_data_received_256

Element type counter

Table 597. 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 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.

Number of Statements with Outbound Bytes Sent Between 257 and 512 Bytes

Element identifier max_data_sent_512

Element type counter

Table 598. 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 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.

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

Element identifier max_data_received_512

Element type counter

Table 599. 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 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.

Number of Statements with Outbound Bytes Sent Between 513 and 1024 Bytes

Element identifier max_data_sent_1024

Element type counter

Table 600. 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 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.

Number of Statements with Outbound Bytes Received Between 513 and 1024 Bytes

Element identifier max_data_received_1024

Element type counter

Table 601. 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 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.

Number of Statements with Outbound Bytes Sent Between 1025 and 2048 Bytes

Element identifier max_data_sent_2048

Element type counter

Table 602. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement

Table 602. 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 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.

Number of Statements with Outbound Bytes Received Between 1025 and 2048 Bytes

Element identifier max_data_received_2048

Element type counter

Table 603. 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 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.

Number of Statements with Outbound Bytes Sent Between 2049 and 4096 Bytes

Element identifier max_data_sent_4096

Element type counter

Table 604. 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 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.

Number of Statements with Outbound Bytes Received Between 2049 and 4096 Bytes

Element identifier max_data_received_4096

Element type counter

Table 605. 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 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.

Number of Statements with Outbound Bytes Sent Between 4097 and 8192 Bytes

Element identifier max_data_sent_8192

Element type counter

Table 606. 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 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.

Number of Statements with Outbound Bytes Received Between 4097 and 8192 Bytes

Element identifier max_data_received_8192

Element type counter

Table 607. 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 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.

Number of Statements with Outbound Bytes Sent Between 8193 and 16384 Bytes

Element identifier max_data_sent_16384

Element type counter

Table 608. 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 with outbound bytes sent 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.

Number of Statements with Outbound Bytes Received Between 8193 and 16384 Bytes

Element identifier max_data_received_16384

Element type counter

Table 609. 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 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.

Number of Statements with Outbound Bytes Sent Between 16385 and 31999 Bytes

Element identifier max_data_sent_31999

Element type counter

Table 610. 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 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.

Number of Statements with Outbound Bytes Received Between 16385 and 31999 Bytes

Element identifier max_data_received_31999

Element type counter

Table 611. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement

Table 611. 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 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.

Number of Statements with Outbound Bytes Sent Between 32000 and 64000 Bytes

Element identifier max_data_sent_64000

Element type counter

Table 612. 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 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.

Number of Statements with Outbound Bytes Received Between 32000 and 64000 Bytes

Element identifier max_data_received_64000

Element type counter

Table 613. 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 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.

Number of Statements with Outbound Bytes Sent Greater than 64000 Bytes

Element identifier max_data_sent_gt64000

Element type counter

Table 614. 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 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.

Number of Statements with Outbound Bytes Received Greater than 64000 Bytes

Element identifier max_data_received_gt64000

Element type counter

Table 615. 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 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.

Number of Statements with Network Time of up to 2 ms

Element identifier max_network_time_2_ms

Element type counter

Table 616. 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 whose network time was less or equal to 2 milliseconds. (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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Number of Statements with Network Time between 2 and 4 ms

Element identifier max_network_time_4_ms

Element type counter

Table 617. 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 whose network time was greater than 2 milliseconds but less or equal to 4 milliseconds. (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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Number of Statements with Network Time between 4 and 8 ms

Element identifier max_network_time_8_ms

Element type counter

Table 618. 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 whose network time was greater than 4 milliseconds but less or equal to 8 milliseconds. (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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Number of Statements with Network Time between 8 and 16 ms

Element identifier max_network_time_16_ms

Element type counter

Table 619. 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 whose network time was greater than 8 milliseconds but less or equal to 16 milliseconds. (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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Number of Statements with Network Time between 16 and 32 ms

Element identifier max_network_time_32_ms

Element type counter

Table 620. 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 whose network time was greater than 16 milliseconds but less or equal to 32 milliseconds. (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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Number of Statements with Network Time greater than 32 ms

Element identifier max_network_time_gt32_ms

Element type counter

Table 621. 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 whose network time was greater than 32 milliseconds. (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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Maximum Network Time for Statement

Element identifier network_time_top

Element type water mark

Table 622. 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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Minimum Network Time for Statement

Element identifier network_time_bottom

Element type water mark

Table 623. 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 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:

- "Host Response Time" on page 459
- "Elapsed Statement Execution Time" on page 408

Transaction ID

Element identifier xid

Element type information

Table 624. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Application	dcs_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.

Statement Execution Elapsed Time

Element identifier elapsed_exec_time

Element type time

Table 625. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement, Timestamp
DCS Application	dcs_appl	Statement, Timestamp
DCS Statement	dcs_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.

Related reference:

- "Host Response Time" on page 459

Host Response Time

Element identifier host_response_time

Element type time

Table 626. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Database	dcs_dbase	Statement
DCS Application	dcs_appl	Statement, Timestamp
DCS Statement	dcs_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):

```
(outbound bytes sent + outbound bytes received) / host response time
```

Related reference:

- "Outbound Number of Bytes Sent" on page 440
- "Outbound Number of Bytes Received" on page 441
- "Statement Execution Elapsed Time" on page 459

Number of Transmissions

Element identifier num_transmissions

Element type counter

Table 627. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Statement	dcs_stmt	Statement

Description

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.

Most Recent Response Time for Connect

Element identifier con_response_time

Element type time

Table 628. 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:

- "Package Cache Overflows" on page 280

Most Recent Connection Elapsed Time

Element identifier con_elapsed_time

Element type time

Table 629. 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:

- "Package Cache Overflows" on page 280

Communication Errors

Element identifier gw_comm_errors

Element type counter

Table 630. 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:

- "Communication Error Time" on page 462
- "Most Recent Statement Elapsed Time" on page 393

Communication Error Time

Element identifier gw_comm_error_time

Element type timestamp

Table 631. 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:

- "Communication Errors" on page 461

Blocking Cursor

Element identifier blocking_cursor

Element type information

Table 632. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Application	stmt	Statement
DCS Statement	dcs_stmt	Statement

Table 633. 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.

Related reference:

- "Outbound Blocking Cursor" on page 463

Outbound Blocking Cursor

Element identifier outbound_blocking_cursor

Element type information

Table 634. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
DCS Statement	dcs_stmt	Statement

Description

This element indicates whether blocking is used for data transfer from the DRDA server to the DB2 Connect gateway for a particular query.

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.

Related reference:

- "Blocking Cursor" on page 463

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 NT 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:

- TP Monitor Client User ID
- TP Monitor Client Workstation Name
- TP Monitor Client Application Name
- TP Monitor Client Accounting String.

TP Monitor Client User ID

Element identifier tpmon_client_userid

Element type information

Table 635. 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:

- "TP Monitor Client Workstation Name" on page 465
- "TP Monitor Client Application Name" on page 465
- "TP Monitor Client Accounting String" on page 466

TP Monitor Client Workstation Name

Element identifier tpmon_client_wkstn

Element type information

Table 636. 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:

- "TP Monitor Client User ID" on page 464
- "TP Monitor Client Application Name" on page 465
- "TP Monitor Client Accounting String" on page 466

TP Monitor Client Application Name

Element identifier tpmon_client_app

Element type information

Table 637. 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:

- "TP Monitor Client User ID" on page 464
- "TP Monitor Client Workstation Name" on page 465
- "TP Monitor Client Accounting String" on page 466

TP Monitor Client Accounting String

Element identifier tpmon_acc_str

Element type information

Table 638. 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:

- "TP Monitor Client User ID" on page 464
- "TP Monitor Client Workstation Name" on page 465
- "TP Monitor Client Application Name" on page 465

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:

- Data Source Name
- Disconnects
- Inserts
- Updates
- Deletes
- Create Nicknames
- Pass-Through
- Stored Procedures

- Remote Locks
- Rows Returned by Stored Procedures
- Query Response Time
- Insert Response Time
- Update Response Time
- Delete Response Time
- Create Nickname Response Time
- Pass-Through Time
- Stored Procedure Time
- Remote Lock Time

Data Source Name

Element identifier datasource_name

Element type information

Table 639. 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

Element identifier disconnects

Element type counter

Table 640. 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.

Inserts

Element identifier insert_sql_stmts

Element type counter

Table 641. 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 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)
```

Updates

Element identifier update_sql_stmts

Element type counter

Table 642. 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)
```

Deletes

Element identifier delete_sql_stmts

Element type counter

Table 643. 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:

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

Create Nicknames

Element identifier create_nickname

Element type counter

Table 644. 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.

Pass-Through

Element identifier passthrus

Element type counter

Table 645. 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 Procedures

Element identifier stored_procs

Element type counter

Table 646. 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

Element identifier remote_locks

Element type counter

Table 647. 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.

Rows Returned by Stored Procedures

Element identifier sp_rows_selected

Element type counter

Table 648. 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:

```
rows per stored proc. = rows returned / # of stored procs. invoked
```

You can also compute the average time to return a row to the federated server from the data source for this application:

```
average time = aggregate stored proc. response time / rows returned
```

Query Response Time

Element identifier select_time

Element type counter

Table 649. Snapshot Monitoring Information

Snapshot Level	Logical Data Grouping	Monitor Switch
Database	dbase_remote	Timestamp

Table 649. Snapshot Monitoring Information (continued)

Snapshot Level	Logical Data Grouping	Monitor Switch
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 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 Response Time

Element identifier insert_time

Element type counter

Table 650. 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 Response Time

Element identifier update_time

Element type counter

Table 651. 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 Response Time

Element identifier delete_time

Element type counter

Table 652. 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 Response Time

Element identifier create_nickname_time

Element type counter

Table 653. 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.

Pass-Through Time

Element identifier passthru_time

Element type counter

Table 654. 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 Procedure Time

Element identifier stored_proc_time

Element type counter

Table 655. 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

Element identifier remote_lock_time

Element type counter

Table 656. 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.

Chapter 7. Monitor Interfaces

Database system monitor interfaces

Monitoring task	API
Capturing a snapshot	db2GetSnapshot - Get Snapshot
Converting the self-describing data stream	db2ConvMonStream - Convert Monitor Stream
Displaying the database system monitor switches	db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot Output Buffer
Estimating the size of a snapshot	db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot Output Buffer
Get/update monitor switches	db2MonitorSwitches - Get/Update Monitor Switches
Resetting monitor counters	db2ResetMonitor - Reset Monitor
Updating the database system monitor switches	db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot Output Buffer

Monitoring task	CLP Command
Analyzing event monitor output with a GUI tool	db2eva - Event Analyzer Command
Capturing a snapshot	GET SNAPSHOT Command
Displaying the database manager monitor switches	GET DATABASE MANAGER MONITOR SWITCHES Command
Displaying the monitoring application's monitor switches	GET MONITOR SWITCHES Command
Formatting the event monitor trace	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
Listing the applications connected to a database	LIST APPLICATIONS Command
Listing the DCS applications	LIST DCS APPLICATIONS Command
Resetting monitor counters	RESET MONITOR Command
Updating the database system monitor switches	UPDATE MONITOR SWITCHES Command

Database system monitor interfaces

Monitoring task	SQL Statement
Activating an event monitor	SET EVENT MONITOR STATE statement
Creating an event monitor	CREATE EVENT MONITOR statement
Deactivating an event monitor	SET EVENT MONITOR STATE statement
Removing an event monitor	DROP statement
Writing event monitor values	FLUSH EVENT MONITOR statement

Monitoring task	SQL Function
Determining the state of an event monitor	EVENT_MON_STATE scalar function
Getting a database manager level snapshot	SNAPSHOT_DBM
Getting the current monitor switch settings at the database manager level	SNAPSHOT_SWITCHES
Getting a fast communication manager snapshot	SNAPSHOT_FCM
Getting a fast communication manager snapshot for a given partition	SNAPSHOT_FCM_NODE
Getting a database level snapshot	SNAPSHOT_DATABASE
Getting an application level snapshot	SNAPSHOT_APPL
Getting an application level snapshot	SNAPSHOT_APPL_INFO
Getting an application level snapshot for lock wait information	SNAPSHOT_LOCKWAIT
Getting an application level snapshot for statement information	SNAPSHOT_STATEMENT
Getting an application level snapshot for agent information	SNAPSHOT_AGENT
Getting an application level snapshot for subsection information	SNAPSHOT_SUBSECT
Getting a buffer pool level snapshot	SNAPSHOT_BP
Getting a table space level snapshot	SNAPSHOT_TBS
Getting a table space level snapshot for configuration information	SNAPSHOT_TBS_CFG
Getting a table space level snapshot for container information	SNAPSHOT_TBS_CONTAINER
Getting a table space level snapshot for quiescer information	SNAPSHOT_QUIESCER

Monitoring task	SQL Function
Getting a table space level snapshot for the ranges of a table space map	SNAPSHOT_RANGES
Getting a table level snapshot	SNAPSHOT_TABLE
Getting a lock level snapshot	SNAPSHOT_LOCK
Getting a snapshot of SQL statement cache information	SNAPSHOT_DYN_SQL

Related reference:

- "EVENT_MON_STATE scalar function" in the *SQL Reference, Volume 1*
- "CREATE EVENT MONITOR statement" in the *SQL Reference, Volume 2*
- "SET EVENT MONITOR STATE statement" in the *SQL Reference, Volume 2*
- "SYSCAT.EVENTMONITORS catalog view" in the *SQL Reference, Volume 1*
- "SYSCAT.EVENTS catalog view" in the *SQL Reference, Volume 1*
- "db2GetSnapshot - Get Snapshot" in the *Administrative API Reference*
- "db2MonitorSwitches - Get/Update Monitor Switches" in the *Administrative API Reference*
- "db2GetSnapshotSize - Estimate Size Required for db2GetSnapshot Output Buffer" in the *Administrative API Reference*
- "db2ResetMonitor - Reset Monitor" in the *Administrative API Reference*
- "GET SNAPSHOT Command" in the *Command Reference*
- "GET MONITOR SWITCHES Command" in the *Command Reference*
- "GET DATABASE MANAGER MONITOR SWITCHES Command" in the *Command Reference*
- "LIST APPLICATIONS Command" in the *Command Reference*
- "LIST DCS APPLICATIONS Command" in the *Command Reference*
- "RESET MONITOR Command" in the *Command Reference*
- "LIST ACTIVE DATABASES Command" in the *Command Reference*
- "db2eva - Event Analyzer Command" in the *Command Reference*
- "db2evmon - Event Monitor Productivity Tool Command" in the *Command Reference*
- "FLUSH EVENT MONITOR statement" in the *SQL Reference, Volume 2*
- "db2ConvMonStream - Convert Monitor Stream" in the *Administrative API Reference*

Part 3. Appendixes

Appendix A. Health Center

Health Indicators

Health indicators are used by the Health Monitor to evaluate specific aspects of database manager or database performance. A health indicator is a specific measurement that gauges the healthiness of some aspect of a particular class of database objects, for example table spaces. Health indicators measure either a finite set of distinct states or a continuous range of values to determine whether the state is healthy or unhealthy. The state defines whether or not the database object or resource is operating normally. If the change in the state is determined to be unhealthy, the Health Monitor issues an alert through the specified reporting channels.

An alert is generated in response to either a change to a non-normal state or the value of the health indicator falling into a warning or alarm zone based on defined threshold boundaries. For health indicators measuring distinct states, an attention type 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. If, for example, the value enters the threshold range of values that defines an alarm zone, an alarm type alert is issued to indicate that the problem need immediate attention. There are three types of alerts: alarm, warning, and attention.

Health Monitor information can be accessed through the Health Center, Web Health Center, the CLP, or APIs. Health indicator configuration is available through these same tools.

The following are the categories of health indicators:

- Table space storage
- Sorting
- Database management system
- Database
- Logging
- Application concurrency
- Package and catalog caches, and workspaces
- Memory

Related concepts:

- "Database system monitor" on page 3
- "Database system monitor data organization" on page 4
- "Tools for monitoring the health of your systems" in the *What's New*

Table space storage health indicators

Table Space Utilization - ts.utilization

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.
- The indicator is calculated using the formula:

  ```
  (ts.used / ts.useable) * 100
  ```

  where ts.used and ts.useable are the system monitor monitor elements Used Pages in Table Space and Useable Pages in Table Space.
- 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.

Additional Information:

- Display string containing information about the short term growth rate, the long term growth rate, and time remaining to fullness.

Resolution:

- Investigate storage utilization

  To analyze a comprehensive view of table space storage utilization, launch the Storage Management tool. This tool can be configured to take storage snapshots as the basis of trend analysis.

  From the Control Center:

  1. Expand the object tree until you find the Table Spaces folder under the parent database for the table space that you want, and select the folder.
  2. Right-click the table space that you want on the detail view, and click Manage Storage... and the Storage Management dialog opens.

3. If there are no snapshots available for the table space or parent database, the Storage Management Setup Launchpad will open before the Storage Management dialog.

- Add a new stripe set

  You can add a new container without rebalancing using the Stripe Sets option. This is recommended over adding containers because adding new containers when the table space is full and containers are the same size forces the table spaces to be rebalanced. With large table spaces, rebalancing can take hours or even days to complete, which can have a significant impact on performance.

  From the Control Center:

  1. Expand the object tree until you find the Table Spaces folder under the parent database for the table space that you want, and select the folder.
  2. Right-click the table space, and click Alter from the pop-up menu. The Alter Table Space notebook opens.
  3. Click on the Containers page. Select the Manage Stripe Set option then select Add to add a new stripe set to the new container.
  4. Complete the required fields on the Define Container dialog and click OK to add the new container. The Define Container dialog closes, returning focus to the Containers page of the Alter Table Space notebook.

  From the Command Line Processor, type as shown in the following example:

  ```
  ALTER TABLESPACE TS1
     BEGIN NEW STRIPE SET (FILE 'FILE3' 100, DEVICE '/dev/rdisk3' 200)
  ```

  where:

  TS1 The name of the table space.

  BEGIN NEW STRIPE SET
  Specifies that a new stripe set is being created in the table space and that containers are being added to this new stripe set.

  (FILE 'FILE3' 100, DEVICE '/dev/rdisk3' 200)
  The name of the container and the container path.

- Add new table space containers

  Increase the number of table space containers to accommodate the anticipated rate of growth. You can use the short and/or long term growth rates available in the additional information for the health indicator to estimate how much capacity you'll need during time period t and use the following formula:

  ```
  t * (growth rate in bytes/time unit)
  ```

Adding new containers when the table space is full, and containers are the same size, forces the table spaces to be rebalanced. With large table spaces, rebalancing can take hours or even days to complete, which can have a significant impact on performance.

From the Control Center:

1. Expand the object tree until you find the Table Spaces folder under the parent database for the table space that you want, and select the folder.
2. Right-click the table space that you want on the detail view, and click Alter from the pop-up menu. The Alter Table Space notebook opens.
3. Click on the Containers page. Select Add to open the Define Container dialog.
4. Complete the required fields on the Define Container dialog and click OK to add the new container. The Define Container dialog closes, returning focus to the Containers page of the Alter Table Space notebook.

Related reference:

- "Health Indicators" on page 485
- "Table Space Container Utilization - tsc.utilization" on page 488
- "Table Space Operational State - ts.state" on page 489
- "Table Space Container Operational State - tsc.state" on page 495

Table Space Container Utilization - tsc.utilization

Description:

- This health indicator tracks the consumption of storage for each SMS table space. An SMS table space is considered full if there is no more space on any of the filesystems for which containers are defined.
- If free space is not available on the filesystem to expand an SMS container, the associated table space becomes full.
- An alert may be issued for each container defined on the filesystem that is running out of free space.
- The indicator is calculated using the formula:

  ```
  (fs.used / fs.total)*100
  ```

  where `fs` is the filesystem 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.

Additional Information:

- Display string containing information about the short term growth rate, the long term growth rate, and time remaining to fullness.

Resolution:

- Investigate storage utilization

  To analyze a comprehensive view of table space storage utilization, launch the Storage Management tool. This tool can be configured to take storage snapshots as the basis of trend analysis.

  From the Control Center:

  1. Expand the object tree until you find the Table Spaces folder under the parent database for the table space that you want, and select the folder.
  2. Right-click the table space that you want on the detail view, and click Manage Storage... and the Storage Management dialog opens.
  3. If there are no snapshots available for the table space or parent database, the Storage Management Setup launchpad will launch before the Storage Management dialog does.
- Free storage on one or more filesystems

  There are two options to free storage:

  1. Extend the size of the filesystem(s) identified in the additional information section based on the growth rate statistics. This option will provide more long-term benefit.
  2. Free space on the filesystem(s) identified in the additional information section. This option is a very short-term remedial action.

Related reference:

- "Health Indicators" on page 485
- "Table Space Utilization - ts.utilization" on page 486
- "Table Space Operational State - ts.state" on page 489
- "Table Space Container Operational State - tsc.state" on page 495

Table Space Operational State - ts.state

Description:

- This health indicator tracks the activity or tasks being performed on a table space. The state of a table space can restrict activity or tasks that can be performed on that table space.

- If the state is non-normal, an Attention alert may be generated to indicate that there is a change from the normal state to another state. The non-normal state may be informational or critical.

Resolution:

- Reset quiesce state (for Quiesced: SHARE state)

  The table space has been put into QUIESCED SHARE state by an explicit request from a user with the appropriate authority. The parent table for the table space cannot be updated with this table space in QUIESCED SHARE state. However, other share mode requests to the table and table spaces are allowed.

  An explicit state reset by the user who issued the original quiesce returns the table space to normal state.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Expand the database and select the Tables folder.
  3. Select the table that holds the table space that you want to quiesce.
  4. Right-click the table, and click Quiesce... from the pop-up menu. The Quiesce dialog opens.
  5. Click on the Quiesce reset radio button. Click the OK button to issue the quiesce reset command.

  From the Command Line Processor, type as shown in the following example:

  ```
  QUIESCE TABLESPACES FOR TABLE tablename RESET
  ```

- Reset quiesce state (for Quiesced: UPDATE state)

  The table space has been put into intent to update mode by an explicit request from a user with the appropriate authority. The table space is locked in intent exclusive (IX) mode and the parent table is locked in update (U) mode.

  An explicit state reset by the user who issued the original quiesce returns the table space to normal state.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Expand the database and select the Tables folder.
  3. Select the table that holds the table space that you want to quiesce.
  4. Right-click the table, and click Quiesce... from the pop-up menu. The Quiesce dialog opens.

5. Click on the Quiesce reset radio button. Click the OK button to issue the quiesce reset command.

From the Command Line Processor, type as shown in the following example:

```
QUIESCE TABLESPACES FOR TABLE tablename RESET
```

- Reset quiesce state (for Quiesced: EXCLUSIVE state)

  The table space has been put into QUIESCED EXCLUSIVE state by an explicit request from a user with the appropriate authority. The user who invoked the quiesce function (the quiescer) has exclusive access to the parent table and this table space. No other access to the table space is allowed.

  An explicit state reset by the user who issued the original quiesce returns the table space to normal state.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Expand the database and select the Tables folder.
  3. Select the table that holds the table space that you want to quiesce.
  4. Right-click the table, and click Quiesce... from the pop-up menu. The Quiesce dialog opens.
  5. Click on the Quiesce reset radio button. Click the OK button to issue the quiesce reset command.

  From the Command Line Processor, type as shown in the following example:

  ```
  QUIESCE TABLESPACES FOR TABLE tablename RESET
  ```

- Not applicable (For Load pending state) The table space has been placed in this state by a vendor's load utility. No table space access is allowed in Load pending state. To remove this state, refer to your load utility vendor's information.
- Not applicable (For Delete pending state) This state is no longer used.
- Backup table space (For Backup pending state)

  The table space has been placed in backup pending state by the load utility because the load process has completed, and:

  – The database configuration parameter logretain is set to recovery, or userexit is enabled, and
  – The load option COPY YES is not specified, and
  – The load option NONRECOVERABLE is not specified.

  The table space is still available for read access. To leave backup pending state, take a backup of the table space.

From the Control Center:

1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
2. Expand the database and select the Table spaces folder. Select the table space that you want from the details view.
3. Right-click the table space, and click Backup... from the pop-up menu. The Backup Table Space dialog opens.
4. Select the media type and complete the mandatory fields in the dialog. Click OK to backup the table space.

From the Command Line Processor, type as shown in the following example:

```
BACKUP DATABASE parent-db-name TABLESPACE (table-space-name)
```

- Rollforward table space (For Rollforward pending state)

  The table space has been placed in rollforward pending state after a table space backup was restored or if the table space was taken offline by the database due to a media error.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Expand the database and select the Table spaces folder. Select the table space that you want from the details view.
  3. Right-click the table space, and click Rollforward... from the pop-up menu. The Roll-Forward Table Space dialog opens.
  4. Complete the mandatory fields in the dialog and click OK to rollforward the table space.

  From the Command Line Processor, type as shown in the following example:

  ```
  ROLLFORWARD DATABASE parent-db-name TABLESPACE (table-space-name)
  ```

- Restore table space (For Restore pending state)

  The table space has been placed in restore pending state for one of two reasons. It could be done by the load utility because a rollforward operation was done through a LOAD command with the COPY NO option specified and the load operation completed successfully. Or, during a rollforward, a log record indicating that a PIT (point in time) rollforward tool place in the past was encountered so the corresponding table space backup has to be restored. To leave restore pending state, perform a restore operation.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.

2. Expand the database and select the Table spaces folder. Select the table space that you want from the details view.
3. Right-click the table space, and click Restore... from the pop-up menu. The Restore Table Space dialog opens.
4. Complete the mandatory fields in the dialog and click OK to restore the table space.

From the Command Line Processor, type as shown in the following example:

```
RESTORE DATABASE parent-db-name TABLESPACE (table-space-name)
```

- Not applicable (For Disable pending state)

  DPS has placed this table space in the disable pending state because it hit an error during a database rollforward. You have to wait for the rollforward to complete for the table space to leave this state.
- Not applicable (For Rollforward in progress state)

  The rollforward utility is in progress on the table space due to an explicit request from the administrator. You have to wait for the utility to complete for the table space to return to normal state.
- Not applicable (For Reorg in progress state)

  The reorg utility is in progress on the table space due to an explicit request from the administrator. You have to wait for the utility to complete for the table space to return to normal state.
- Not applicable (For Backup in progress state)

  The backup utility is in progress on the table space due to an explicit request from the administrator. You have to wait for the utility to complete for the table space to return to normal state.
- Not applicable (For Restore in progress state)

  The restore utility is in progress on the table space due to an explicit request from the administrator. You have to wait for the utility to complete for the table space to return to normal state.
- Not applicable (For Storage must be defined state)

  The table space has been placed in Storage must be defined state because while defining containers for the table space during a redirected restore via the SET TABLESPACE CONTAINERS command, an error occurred while attempting to acquire the specified containers. To remove this state, reissue the SET TABLESPACE CONTAINERS command ensuring that all the containers specified are accessible.
- Place table space online (For Offline and not accessible state)

  The table space is offline or not accessible due to a failure, possibly in table space initialization or rollforward processing. For example, with circular logging, if a table space container is not available on start up, and the database is in a consistent state, the table space will be placed in the offline

state. If the database is in an inconsistent state, the database cannot be restarted until the table space is placed in the DROP PENDING state by using the DROP PENDING TABLESPACES clause in the RESTART DATABASE command. The same situation can occur if logretain is on. However, for the case where the database is in an inconsistent state, the table space is also placed in ROLLFORWARD PENDING STATE.

If the problematic container becomes accessible, then the table space can be placed online either by disconnecting and reconnecting to the database or alter the table space using the SWITCH ONLINE option.

From the Control Center:

1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
2. Expand the database and select the Table spaces folder. Select the table space that you want from the details view.
3. Right-click the table space, and click Alter... from the pop-up menu. The Alter Table Space dialog opens.
4. Click the Automatically put table space online when containers become available check box. Click OK to alter the table space.

From the Command Line Processor, type as shown in the following example:

```
ALTER TABLESPACE table-space-name SWITCH ONLINE
```

- Drop table space (For Drop pending state)

  The table space has been placed in drop pending state because a user requested the state through the DROP PENDING TABLESPACES clause in RESTART DATABASE. This state cannot be removed. The only allowable action is to drop the table space.

  To drop a table space from the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Expand the database and select the Table spaces folder. Select the table space that you want from the details view.
  3. Right-click the table space, and click Drop from the pop-up menu. The Drop Confirmation dialog opens.
  4. Click OK to drop the table space.

  From the Command Line Processor, type as shown in the following example:

  ```
  DROP TABLESPACE table-space-name
  ```

- Not applicable (For Storage may be defined state)

The table space has been placed in this state because a user requested that a redirected restore take place, and now is the opportunity for the user to redefine containers for this table space. To remove this state, continue the restore.

- Not applicable (For DMS rebalancer is active)

  The rebalancer is currently active on this table space. You have to wait for the rebalancer to complete for the table space to return to normal state.

- Not applicable (For TBS creation in progress state)

  The table space was placed in TBS creation in progress state because the table space, although created, is not ready for use. You have to wait for creation to complete for the table space to return to normal state.

- Not applicable (For TBS deletion in progress state)

  The table space was placed in TBS deletion in progress state because the table space deletion has not completed. There is no action to take to remove this state.

- Set write resume for database (For Write suspend state)

  The table space has been placed in this state due to an explicit request from a user by issuing a SET WRITE SUSPEND FOR DATABASE command. To remove this state, set write resume for the database.

  From the Command Line Processor, type as shown in the following example:

  ```
  SET WRITE RESUME FOR DATABASE parent-db-name
  ```

Related reference:

- "Health Indicators" on page 485
- "Table Space Utilization - ts.utilization" on page 486
- "Table Space Container Utilization - tsc.utilization" on page 488
- "Table Space Container Operational State - tsc.state" on page 495

Table Space Container Operational State - tsc.state

Description:

- Tracks the activity being performed on a table space container. The state of a table space container can restrict activity or tasks that can be performed on that table space container.
- If the state is not accessible, an Attention alert may be generated.

Resolution:

- Place table space online (For Not Accessible state)

  The container has been placed in not accessible state because DB2 was unable to open the container while initializing the table space for use. This could happen because the permissions are incorrect or the because the

physical container has been deleted. Ensure that the container exists and the permissions are correct and either, disconnect and reconnect to the database or alter the table space using the SWITCH ONLINE option.

From the Control Center:

1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
2. Expand the database and select the Table spaces folder. Select the table space that you want from the details view.
3. Right-click the table space, and click Alter... from the pop-up menu. The Alter Table Space dialog opens.
4. Click the Automatically put table space online when containers become available check box. Click OK to alter the table space.

From the Command Line Processor, type as shown in the following example:

```
ALTER TABLESPACE table-space-name SWITCH ONLINE
```

Related reference:

- "Health Indicators" on page 485
- "Table Space Utilization - ts.utilization" on page 486
- "Table Space Container Utilization - tsc.utilization" on page 488
- "Table Space Operational State - ts.state" on page 489

Sorting health indicators

Private Sort Memory Utilization - db2.sort_privmem_util

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) >= 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.

Additional Information:

A display string containing the values of db2.post_threshold_sorts, db2.max_priv_sort_mem, and memory usage trend analysis.

Resolution:

- Increase the sort heap threshold

  If private memory is available, increase the database manager configuration parameter SHEAPTHRES to allow for a larger sort heap. Set the new value of SHEAPTHRES to be 100% of the system monitor element db2.max_priv_sort_mem.

  From the Control Center:

  1. Expand the object tree until you find the Instances folder and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click Configure... from the pop-up menu. The DBM Configuration dialog opens.
  3. Expand the Performance category, then update the sort heap threshold parameter and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE MANAGER CONFIGURATION USING SHEAPTHRES size
  ```

- Tune workload

  You can run the Design Advisor to tune the database performance for your workload by adding indexes and materialized query tables. This can help reduce the need for sorting. You will need to provide your query workload and database name. The Design Advisor will evaluate the existing indexes and materialized query tables in terms of the workload and recommend any new objects required.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Optimize for Workload Performance... from the pop-up menu. The Design Advisor opens.
  3. Follow the steps of the Design Advisor to optimize query performance and apply the recommendations of the Design Advisor.

  This function is also available from the command line by typing:

```
db2advis
```

- Increase sort concurrency

  If the database configuration parameter sortheap is bigger than it needs to be, then lower it to get more concurrent sorts in under the thresholds. Update the sortheap value to 100% of the current value of the system monitor element db2.max_priv_sort_mem. The sortheap memory utilization frequency statistic, included in the additional information for this health indicator, provides an indication of behavior for sortheap usage. You want to decrease the value of sortheap to a point where a normal workload is possible without exceeding SHEAPTHRES, but not to the point where performance is seriously impacted.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Performance tab, update the sort heap parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE CONFIGURATION FOR DATABASE name USING SORTHEAP size
  ```

Related reference:

- "Health Indicators" on page 485
- "Long Term Shared Sort Memory Utilization - db.max_sort_shrmem_util" on page 503
- "Shared Sort Memory Utilization - db.sort_shrmem_util" on page 498
- "Percentage of Sorts That Overflowed - db.spilled_sorts" on page 500
- "Long Term Private Sort Memory Utilization - db2.max_sort_privmem_util" on page 502

Shared Sort Memory Utilization - db.sort_shrmem_util

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. Please 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.

Additional Information:

A display string containing the values of db2.max_shr_sort_mem, sortheap, db2.max_priv_sort_mem, and memory usage trend analysis.

Resolution:

- Increase the shared sort heap threshold

  If shared memory is available, increase the shared sortheap threshold database configuration parameter to allow for a larger sort heap. Set the new value of the shared sortheap threshold configuration parameter to be at least 100% of the system monitor element db.max_shr_sort_mem.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Performance tab, update the shared sort heap threshold parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE CONFIGURATION FOR DB name USING SHEAPTHRES_SHR size
  ```

  Or if SHEAPTHRES_SHR is set to 0 and you want to continue to use SHEAPTHRES for both shared and private sort heap thresholds:

  ```
  UPDATE DATABASE CONFIGURATION FOR DB name USING SHEAPTHRES size
  ```

- Tune workload

  You can run the Design Advisor to tune the database performance for your workload by adding indexes and materialized query tables. This can help reduce the need for sorting. You will need to provide your query workload and database name. The Design Advisor will evaluate the existing indexes and materialized query tables in terms of the workload and recommend any new objects required.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.

2. Right-click the database, and click Optimize for Workload Performance... from the pop-up menu. The Design Advisor opens.
3. Follow the steps of the Design Advisor to optimize query performance and apply the recommendations of the Design Advisor.

This function is also available from the command line by typing:

```
db2advis
```

- Increase sort concurrency

  If the database configuration parameter sortheap is bigger than it needs to be, then lower it to get more concurrent sorts in under the thresholds. Update the sortheap value to 100% of the current value of the system monitor element db2.max_priv_sort_mem. The sortheap memory utilization frequency statistic, included in the additional information for this health indicator, provides an indication of behavior for sortheap usage. You want to decrease the value of sortheap to a point where a normal workload is possible without exceeding SHEAPTHRES, but not to the point where performance is seriously impacted.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Performance tab, update the sort heap parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE CONFIGURATION FOR DATABASE name USING SORTHEAP size
  ```

Related reference:

- "Health Indicators" on page 485
- "Long Term Shared Sort Memory Utilization - db.max_sort_shrmem_util" on page 503
- "Private Sort Memory Utilization - db2.sort_privmem_util" on page 496
- "Percentage of Sorts That Overflowed - db.spilled_sorts" on page 500
- "Long Term Private Sort Memory Utilization - db2.max_sort_privmem_util" on page 502

Percentage of Sorts That Overflowed - db.spilled_sorts

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: (db.sort_overflows / db.total_sorts)*100. The system monitor element db.sort_overflows 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 is the total number of sorts that have been executed.

Additional Information:

A display string containing the values of db2.max_priv_sort_mem and sortheap.

Resolution:

- Increase sort heap

  Increase the database configuration parameter sortheap so that spilling won't occur unnecessarily. Update the sortheap value to 100% of the current value of the snapshot element db2.max_priv_sort_mem. The sortheap memory utilization frequency statistic, included in the additional information for this health indicator, provides an indication of behavior on this attribute.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Performance tab, update the sort heap parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE CONFIGURATION FOR DATABASE name USING SORTHEAP size
  ```

- Tune workload

  You can run the Design Advisor to tune the database performance for your workload by adding indexes and materialized query tables. This can help reduce the need for sorting. You will need to provide your query workload and database name. The Design Advisor will evaluate the existing indexes and materialized query tables in terms of the workload and recommend any new objects required.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.

2. Right-click the database, and click Optimize for Workload Performance... from the pop-up menu. The Design Advisor opens.
3. Follow the steps of the Design Advisor to optimize query performance and apply the recommendations of the Design Advisor.

This function is also available from the command line by typing:

```
db2advis
```

Related reference:

- "Health Indicators" on page 485
- "Long Term Shared Sort Memory Utilization - db.max_sort_shrmem_util" on page 503
- "Private Sort Memory Utilization - db2.sort_privmem_util" on page 496
- "Shared Sort Memory Utilization - db.sort_shrmem_util" on page 498
- "Long Term Private Sort Memory Utilization - db2.max_sort_privmem_util" on page 502

Long Term Private Sort Memory Utilization - db2.max_sort_privmem_util

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 may be generated when the percentage usage is low.
- The indicator is calculated using the formula: (db2.max_priv_sort_mem / sheapthres)*100.

Additional Information:

A display string containing the values of db2.max_priv_sort_mem.

Resolution:

- Decrease the sort heap threshold

  Decrease the database manager configuration parameter SHEAPTHRES because it is currently too big. Set the new value of SHEAPTHRES to be at least 100% of db2.max_priv_sort_mem.

  From the Control Center:

  1. Expand the object tree until you find the Instances folder and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click Configure... from the pop-up menu. The DBM Configuration dialog opens.

3. Expand the Performance category, then update the sort heap threshold parameter and click OK to apply the update.

From the Command Line Processor, type as shown in the following example:

```
UPDATE DATABASE MANAGER CONFIGURATION USING SHEAPTHRES size
```

Related reference:

- "Health Indicators" on page 485
- "Long Term Shared Sort Memory Utilization - db.max_sort_shrmem_util" on page 503
- "Private Sort Memory Utilization - db2.sort_privmem_util" on page 496
- "Shared Sort Memory Utilization - db.sort_shrmem_util" on page 498
- "Percentage of Sorts That Overflowed - db.spilled_sorts" on page 500

Long Term Shared Sort Memory Utilization - db.max_sort_shrmem_util

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 may be generated when the percentage usage is low.
- The indicator is calculated using the formula: (db.max_shr_sort_mem / sheapthres_shr)*100.

Additional Information:

A display string containing the values of db2.max_priv_sort_mem and sortheap.

Resolution:

- Decrease the shared sort heap threshold

  Decrease the shared sortheap threshold database configuration parameter because it is currently too big. Set the new value of the shared sortheap threshold configuration parameter to be db.max_shr_sort_mem * 110.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.

3. On the Performance tab, update the shared sort heap threshold parameter as recommended above and click OK to apply the update.

From the Command Line Processor, type as shown in the following example:

```
UPDATE DATABASE CONFIGURATION FOR DB name USING SHEAPTHRES_SHR size
```

Or if SHEAPTHRES_SHR is set to 0 and you want to continue to use SHEAPTHRES for both shared and private sort heap thresholds:

```
UPDATE DATABASE CONFIGURATION FOR DB name USING SHEAPTHRES size
```

Related reference:

- "Health Indicators" on page 485
- "Private Sort Memory Utilization - db2.sort_privmem_util" on page 496
- "Shared Sort Memory Utilization - db.sort_shrmem_util" on page 498
- "Percentage of Sorts That Overflowed - db.spilled_sorts" on page 500
- "Long Term Private Sort Memory Utilization - db2.max_sort_privmem_util" on page 502

Database manager health indicators

Instance Operational State - db2.db2_op_status

Description:

- An instance is considered healthy if the instance state does not restrict activity or tasks being performed.
- This indicator tracks the state of the instance. A non-Active state may generate an Attention alert.

Resolution:

- Unquiesce the instance (For Quiesce Pending state)

  The instance is in quiesce pending state due to an explicit request by the administrator. The instance won't allow attachments. To return the instance to active state, you can issue an unquiesce.

  From the Control Center:

  1. Expand the object tree until you find the Instances folder and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click Unquiesce from the pop-up menu.

  From the Command Line Processor, type as shown in the following example:

  ```
  UNQUIESCE INSTANCE instance-name
  ```

- Unquiesce the instance (For Quiesce state)

  The instance is in quiesced state due to an explicit request by the administrator. The instance won't allow attachments. To return the instance to active state, you can issue an unquiesce.

  From the Control Center:

  1. Expand the object tree until you find the Instances folder and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click Unquiesce from the pop-up menu.

  From the Command Line Processor, type as shown in the following example:

  ```
  UNQUIESCE INSTANCE instance-name
  ```

- Start the instance (For Down state)

  The instance is not active due to an explicit stop request by the administrator or abnormal termination of the instance. Start the instance to return it to active state or configure Fault Monitor to keep the instance highly available.

  To start the instance from the Control Center:

  1. Expand the object tree until you find the Instances folder and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click Start from the pop-up menu.

Related reference:

- "Health Indicators" on page 485
- "DBMS Highest Severity Alert State - db2.db2_alert_state" on page 505

DBMS Highest Severity Alert State - db2.db2_alert_state

Description:

- 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.

Resolution:

- Not applicable.

Related reference:

- "Health Indicators" on page 485
- "Instance Operational State - db2.db2_op_status" on page 504

Database health indicators

Database Operational State - db.db_op_status

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.

Resolution:

- Unquiesce the database (For Quiesce Pending state)

  The database has been put into quiesce-pending state by an explicit request from the administrator. If you have QUIESCE_CONNECT authority, or are DBADM or SYSADM, you will still have access to the database and will be able to use it normally. For all other users, new connections to the database aren't permitted and new units of work cannot be started. Also, depending on the quiesce request, active units of work will be allowed to complete or will be rolled back immediately. You can issue an unquiesce to return to active state.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Unquiesce from the pop-up menu.

  From the Command Line Processor, type as shown in the following example:

  ```
  CONNECT TO DATABASE db
  UNQUIESCE DATABASE
  ```

- Unquiesce the database (For Quiesce state)

  The database has been put into quiesce state by an explicit request from the administrator. New connections to the database aren't permitted and new units of work cannot be started. Issue an unquiesce to return to active state.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder, then expand the folder until you find the database that you want.
  2. Right-click the database, and click Unquiesce from the pop-up menu.

From the Command Line Processor, type as shown in the following example:

```
CONNECT TO DATABASE db
UNQUIESCE DATABASE
```

- Not applicable (For Rollforward state)

  A rollforward is in progress on the database due to an explicit request from the administrator. You have to wait for the rollforward to complete for the instance to return to active state.

Related reference:

- "Health Indicators" on page 485
- "Database Highest Severity Alert State - db.alert_state" on page 507

Database Highest Severity Alert State - db.alert_state

Description:

- 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
- The alert state determines the overall health of the database.

Resolution:

- Not applicable.

Related reference:

- "Health Indicators" on page 485
- "Database Operational State - db.db_op_status" on page 506

Logging health indicators

Log Utilization - db.log_utilization

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
```

.

- 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.

Additional Information:

A display string containing the values of logprimary, logsecond, logfilsz, and db.appl_id_oldest_xact.

Resolution:

- Increase log space

  If sufficient resources are available, increase logfilsz, logprimary, and/or logsecond. If you can afford to shut down DB2, determine values for the parameters to meet your needs. If you cannot afford to shut down DB2, just increase logsecond which can be updated dynamically.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Logs tab, update the appropriate log parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE CONFIGURATION FOR DB name USING log parameter size
  ```

  where log parameter is one of the parameters listed above.

- End application(s) with oldest uncommitted transactions

  End the application with the id from db.appl_id_oldest_xact shown in the additional details section.

  From the Control Center:

  1. Expand the object tree until you find the instance that is the parent of the database that has an alert on this health indicator.
  2. Right-click on the instance, and click List Applications.... The List Applications dialog opens.
  3. Select the application that you want to force. Right-click on the application, and click Force from the pop-up menu.

From the Command Line Processor, type as shown in the following example:

```
FORCE APPLICATION (appl handle)
```

where appl handle is the application handle for the application you want to force.

- Resolve indoubt transactions

  If the db.indoubt_trans_exist health indicator is enabled for this database, look at that alert for details on the indoubt transactions. Otherwise, invoke the Indoubt Transaction Manager to check the current indoubt transactions in this database. Usually, the transaction manager will attempt to resynchronize indoubt transactions. However, it is possible that an indoubt transaction may not be resolved automatically. For example, an indoubt transaction may involve more than just your DB2 system. If a third party tool involved in the transaction cannot be reached due to communication failure, or is unavailable, the indoubt transaction may not be resolved automatically. If something like this has happened, or if you cannot wait for the transaction manager to automatically resolve indoubt transactions, there are actions you can take to manually and heuristically resolve indoubt transactions and free up log space.

  The Indoubt Transaction Manager is available from the command line by typing:

  ```
  db2indbt
  ```

Related reference:

- "Health Indicators" on page 485
- "Log Filesystem Utilization - db.log_fs_utilization" on page 509
- "Indoubt Transactions Existence - db.indoubt_trans_exist" on page 510

Log Filesystem Utilization - db.log_fs_utilization

Description:

- Log Filesystem Utilization tracks the fullness of the filesystem on which the transaction logs reside. DB2 may not be able to create a new log file if there is no room on the filesystem.
- Log utilization is measured as the percentage of space consumed. If the amount of free space in the filesystem 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 filesystem 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.

Additional Information:

A display string containing the values of logprimary, logsecond, and logfilsz and user_exit and blk_log_dsk_full.

Resolution:

- Dedicate filesystem to logging

  Check if other processes are using the filesystem on which the logs reside. If so, consider dedicating the filesystem to logging so that utilization of the filesystem is not affected by anything but logging.
- Archive logs

  If userexit has been enabled, see the Additional Information, ensure that the user exit program is functioning normally. If you cannot fix the user exit archiving immediately, consider manually archiving the log files to free storage on the filesystem.

  If userexit has not been enabled, consider enabling it to allow automatic archiving of log files. Otherwise, you can archive the logs manually to free storage on the filesystem.
- Free storage or extend filesystem used for logs

  Extend the size of the filesystem on which the log resides. Alternatively, free up space on the same file system to provide a short term remedy to the problem.
- Investigate log utilization

  If there are current alerts on ts.utilization or tsc.utilization, these alerts need to be resolved first. It is likely that resolving the utilization alerts will make this alert go away. However, you should investigate the file system space to avoid this situation in future.

Related reference:

- "Health Indicators" on page 485
- "Log Utilization - db.log_utilization" on page 507
- "Indoubt Transactions Existence - db.indoubt_trans_exist" on page 510

Indoubt Transactions Existence - db.indoubt_trans_exist

Description:

- This health indicator tracks the number of outstanding indoubt transactions.

- Indoubt transactions hold log space for uncommitted transaction causing logs to become full and preventing further transactions from taking place.
- If indoubt transactions exist, an Attention alert may be generated.

Resolution:

- Resolve indoubt transactions

  Invoke the Indoubt Transaction Manager to check the current indoubt transactions in this database. Usually, the transaction manager will attempt to resynchronize indoubt transactions. However, it is possible that an indoubt transaction may not be resolved automatically. For example, an indoubt transaction may involve more than just your DB2 system. If a third party tool involved in the transaction cannot be reached due to communication failure, or is unavailable, the indoubt transaction may not be resolved automatically. If something like this has happened, or if you cannot wait for the transaction manager to automatically resolve indoubt transactions, there are actions you can take to manually and heuristically resolve indoubt transactions and free up log space.

  The Indoubt Transaction Manager is available from the command line by typing:

  ```
  db2indbt
  ```

Related reference:

- "Health Indicators" on page 485
- "Log Utilization - db.log_utilization" on page 507
- "Log Filesystem Utilization - db.log_fs_utilization" on page 509

Application concurrency health indicators

Deadlock Rate - db.deadlock_rate

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 "<time-amount>" ago.
- The higher the rate of deadlocks, the greater the degree of contention which may generate an alert.

Resolution:

- Investigate current deadlocks

  Use the Event Monitor tool to monitor deadlocks. The Event Monitor is available as a graphical tool in the Control Center. You can start an event monitor to capture deadlock events and then determine which applications are involved in deadlocks.

  From the Control Center:

  1. Expand the object tree until you find the Database folder and expand the folder until you find the database that you want.
  2. Expand the database and select the Event Monitor folder.
  3. Right-click the Event Monitor folder, and click Create.... The Create Event Monitor window opens.
  4. Enter an event monitor name in the Name field. Click on OK to create the deadlock event monitor.
- Tune applications to execute better concurrently

  Some applications may not be capable of running concurrently. If you determine applications in which deadlocks are occurring that are capable of running concurrently, you may be able to modify the application to better enable it to execute concurrently. Contention 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.

Related reference:

- "Health Indicators" on page 485

Lock List Utilization - db.locklist_utilization

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.
- Returns an error 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.

Resolution:

- Increase size of lock list

  If there's not enough lock list space available, lock escalations will take place thereby increasing contention and reducing concurrency. Update locklist to be equal to

  ```
  (db.lock_list_in_use / (4096*U))
  ```

  where U is 60% of the warning threshold level to ensure utilization sufficiently below a warning level. For example, if the warning threshold is 75%, then:

  ```
  U = 0.6 * 0.75 = 0.45 (or 45%).
  ```

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.

3. On the Performance tab, update the lock list parameter as recommended above and click OK to apply the update.

From the Command Line Processor, type as shown in the following example:

```
UPDATE DATABASE CONFIGURATION FOR DB name USING LOCKLIST size
```

- Tune applications to reduce locking

  Use the Database System Monitor application snapshot to identify which applications are holding many locks and then consider the following suggestions for controlling the size of the lock list:

  – Perform frequent COMMITs to release locks.
  – When performing many updates, lock the entire table before updating (using the SQL LOCK TABLE statement) so as to use only one lock and keep others from interfering with updates. This does reduce concurrency of the data.
  – Use the LOCKSIZE parameter of the ALTER TABLE statement to control how locking is done for a specific table. For details, refer to the SQL Reference.
  – Use the Cursor Stability isolation level when possible to decrease the number of share locks held. If applications integrity requirements are not compromised, use Uncommitted Read instead of Cursor Stability to further decrease the amount of locking.

- Decrease percentage of lock list for use by applications

  If you investigate into lock usage by applications and determine that one or a small number of applications are consuming most of the locklist, then you can throttle back those applications. This action should be the last resort, and used only if you can't increase the lock list size or tune applications to not use so many locks. Decreasing the maxlocks database configuration parameter can cause lock escalations.

  Assuming that the maximum number of locks held by applications is M, then maxlocks can be set to (M*36 / locklist) if maxlocks > (M*36 / locklist).

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Applications tab, update the maximum locks parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

```
UPDATE DATABASE CONFIGURATION FOR DB name USING MAXLOCKS size
```

Related reference:

- "Health Indicators" on page 485
- "Deadlock Rate - db.deadlock_rate" on page 511
- "Lock Escalation Rate - db.lock_escal_rate" on page 515
- "Percentage of Applications Waiting on Locks - db.apps_waiting_locks" on page 517

Lock Escalation Rate - db.lock_escal_rate

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 "<time-amount>" ago.
- The higher the rate of deadlocks, the greater the degree of contention which may generate an alert.

Additional Information:

A display string containing the values of locklist, and the amount of locklist memory in use.

Resolution:

- Increase size of lock list

  If there's not enough lock list space available, lock escalations will take place thereby increasing contention and reducing concurrency. Update locklist to be equal to

  ```
  (db.lock_list_in_use / (4096*U))
  ```

where U is 60% of the warning threshold level to ensure utilization sufficiently below a warning level. For example, if the warning threshold is 75%, then U = 0.6 * 0.75 = 0.45 (or 45%).

From the Control Center:

1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
3. On the Performance tab, update the lock list parameter as recommended above and click OK to apply the update.

From the Command Line Processor, type as shown in the following example:

```
UPDATE DATABASE CONFIGURATION FOR DB name USING LOCKLIST size
```

- Increase the percent of lock list usable by each application

  When applications reach their share (MAXLOCKS) of the lock list, then escalations will take place so that the applications stay under the maximum. You can use the Database System Monitor application snapshot to determine which applications are holding a lot of locks at the maxlock value.

  Increase the maxlocks database configuration parameter value to accommodate the largest use as given by the monitor.

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Applications tab, update the maximum locks parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE CONFIGURATION FOR DB name USING MAXLOCKS size
  ```

- Reduce the number of locks used by applications

  One or more applications may be holding an excessive number of locks. Identify the applications holding large numbers of locks using the Database System Monitor application snapshot. You can then:

  1. Perform more commits in these applications
  2. Use table locks instead of row locks in these applications.

Related reference:

- "Health Indicators" on page 485
- "Deadlock Rate - db.deadlock_rate" on page 511
- "Lock List Utilization - db.locklist_utilization" on page 513
- "Percentage of Applications Waiting on Locks - db.apps_waiting_locks" on page 517

Percentage of Applications Waiting on Locks - db.apps_waiting_locks

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.appls_cur_cons) *100)
  ```

Resolution:

- Tune applications to reduce contention

  Use the Database System Monitor application snapshot to identify the applications with the longest wait time for current locks. For those applications holding locks too long, you can:

  – Turn on lock timeout detection. If the lock timeout detection is currently off (i.e. the value of locktimeout is -1), you should turn it on by setting the value of the database configuration parameter Lock timeout (locktimeout) to be greater than -1. This value specifies the number of seconds that an application will wait to obtain a lock. NOTE: If you set this parameter to 0, locks are not waited for. In this situation, if no lock is available at the time of the request, the applications immediately receives a -911. Therefore, we strongly recommend that you do not set the locktime to 0 for this parameter.

  – Reduce the time an application will wait to obtain a lock. If the lock timeout detection is currently on (i.e. the value of locktimeout is greater than -1), you can reduce the time an application will wait to obtain a lock by decreasing the value of the database configuration parameter Lock timeout (locktimeout). NOTE: If you set this parameter to 0, locks are not waited for. In this situation, if no lock is available at the time of the request, the applications immediately receives a -911. Therefore, we strongly recommend that you do not set the locktime to 0 for this parameter.

  – Reduce the time interval for checking deadlock. You can reduce the time interval for checking deadlock by decreasing the value of the database configuration parameter Time Interval for Checking Deadlock (dlchktime). Decreasing this parameter increases the frequency of

checking for deadlocks, thereby decreasing the time that application programs must wait for the deadlock to be resolved but increasing the time that the database manager takes to check for deadlocks. If the deadlock interval is too small, it can decrease run-time performance, because the database manager is frequently performing deadlock detection. If this parameter is set lower to improve concurrency, you should ensure that maxlocks and locklist are set appropriately to avoid unnecessary lock escalation, which can result in more lock contention and as a result, more deadlock situations.

- Increase the maximum storage for lock list. Increase the amount of storage that is allocated to the lock list by increasing the database configuration parameter Maximum Storage for Lock List (locklist). Increasing this value can prevent lock escalations, which decrease concurrency and overall database performance.
- Increase the maximum percent of lock list before escalation. The database configuration parameter, Maximum Percent of Lock List Before Escalation (MAXLOCKS), defines a percentage of the lock list held by an application that must be filled before the database manager performs escalation. You can increase this value to prevent lock escalations, which decreases concurrency and overall database performance.
- Perform frequent COMMITs to release locks. This can help control the size of the lock list, and therefore prevent lock escalations, which decrease concurrency and overall database performance. This also has the side effect of controlling the amount of log space used by a single transaction, allowing the first log sequence number belonging to the oldest active transaction to move up.

Related reference:

- "Health Indicators" on page 485
- "Deadlock Rate - db.deadlock_rate" on page 511
- "Lock List Utilization - db.locklist_utilization" on page 513
- "Lock Escalation Rate - db.lock_escal_rate" on page 515

Package and catalog caches, and workspaces health indicators

Catalog Cache Hit Ratio - db.catcache_hitratio

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-(db.cat_cache_inserts/db.cat_cache_lookups))*100
```

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and directly change the value of the configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that is the parent of the database that you want.
  2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
  3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance and each database. Catalog Cache Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.
  4. Click on the check box on the Show Plot column for the Catalog Cache Heap row to add the element to the plot.

Related reference:

- "Health Indicators" on page 485
- "Package Cache Hit Ratio - db.pkgcache_hitratio" on page 519
- "Shared Workspace Hit Ratio - db.shrworkspace_hitratio" on page 520

Package Cache Hit Ratio - db.pkgcache_hitratio

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-(db.pkg_cache_inserts/db.pkg_cache_lookups))*100
  ```

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and directly change the value of the configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that is the parent of the database that you want.

2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance and each database. Package Cache Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.
4. Click on the check box on the Show Plot column for the Package Cache Heap row to add the element to the plot.

Related reference:

- "Health Indicators" on page 485
- "Catalog Cache Hit Ratio - db.catcache_hitratio" on page 518
- "Shared Workspace Hit Ratio - db.shrworkspace_hitratio" on page 520

Shared Workspace Hit Ratio - db.shrworkspace_hitratio

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-(db.shr_workspace_section_inserts/db.shr_workspace_section_lookups))*100
  ```

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and directly change the value of the configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that is the parent of the database that you want.
  2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
  3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance and each database. Application Control Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.
  4. Click on the check box on the Show Plot column for the Application Control Heap row to add the element to the plot.

Related reference:

- "Health Indicators" on page 485
- "Catalog Cache Hit Ratio - db.catcache_hitratio" on page 518
- "Package Cache Hit Ratio - db.pkgcache_hitratio" on page 519

Memory health indicators

Monitor Heap Utilization - db2.mon_heap_utilization

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:

  ```
  (db2.pool_cur_size / db2.pool_max_size) * 100
  ```

  for the Memory Pool Identifier SQLM_HEAP_MONITOR.
- Once this percentage reaches the maximum, 100%, monitor operations may fail.

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and directly change the value of the configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
  3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance. Monitor Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.
  4. Click on the check box on the Show Plot column for the Monitor Heap row to add the element to the plot.
- Increase the monitor heap size

  Increase the database manager configuration parameter MON_HEAP_SZ sufficiently to move utilization to normal operating levels. Set the new value of MON_HEAP_SZ to be equal to (pool_cur_size / (4096*U)) where U

is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then U = 0.6 * 0.75 = 0.45 (or 45%).

From the Control Center:

1. Expand the object tree until you find the Instances folder and expand the folder until you find the instance that you want.
2. Right-click the instance, and click Configure... from the pop-up menu. The DBM Configuration dialog opens.
3. Expand the Performance category, then update the monitor heap size parameter and click OK to apply the update.

From the Command Line Processor, type as shown in the following example:

```
UPDATE DATABASE MANAGER CONFIGURATION USING MON_HEAP_SZ size
```

Related reference:

- "Health Indicators" on page 485
- "Query Heap Utilization - db2.query_heap_utilization" on page 522
- "Database Heap Utilization - db.database_heap_utilization" on page 524
- "Utility Heap Utilization - db.utility_heap_utilization" on page 525
- "Application Control Heap Utilization - db.applctl_heap_utilization" on page 526
- "Application Heap Utilization - db.appl_heap_utilization" on page 528

Query Heap Utilization - db2.query_heap_utilization

Description:

- Tracks the consumption of the query heap memory, based on the memory pool with the ID SQLM_HEAP_QUERY.
- The utilization is calculated using the formula:

  ```
  (db2.pool_cur_size / db2.pool_max_size) * 100
  ```

  for the Memory Pool Identifier SQLM_HEAP_QUERY.
- Once this percentage reaches the maximum, 100%, query operations may fail and communication may be impacted, particularly for down-level clients.

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and directly change the value of the configuration parameters.

From the Control Center:

1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that you want.
2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance and each database. Query Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.
4. Click on the check box on the Show Plot column for the Query Heap row to add the element to the plot.

- Increase the query heap size

  Increase the database manager configuration parameter QUERY_HEAP_SZ sufficiently to move utilization to normal operating levels. Set the new value of QUERY_HEAP_SZ to be equal to pool_cur_size / (U*4096) where U is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then U = 0.6 * 0.75 = 0.45 (or 45%).

  From the Control Center:

  1. Expand the object tree until you find the Instances folder and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click Configure... from the pop-up menu. The DBM Configuration dialog opens.
  3. Expand the Performance category, then update the query heap size parameter and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  UPDATE DATABASE MANAGER CONFIGURATION USING QUERY_HEAP_SZ size
  ```

Related reference:

- “Health Indicators” on page 485
- “Monitor Heap Utilization - db2.mon_heap_utilization” on page 521
- “Database Heap Utilization - db.database_heap_utilization” on page 524
- “Utility Heap Utilization - db.utility_heap_utilization” on page 525
- “Application Control Heap Utilization - db.applctl_heap_utilization” on page 526
- “Application Heap Utilization - db.appl_heap_utilization” on page 528

Database Heap Utilization - db.database_heap_utilization

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.

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and effect changes directly to configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
  3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance and each database. Database Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.
  4. Click on the check box on the Show Plot column for the Database Heap row to add the element to the plot.
- Increase the database heap size

  Either set the database configuration parameter DBHEAP to automatic, or increase it sufficiently to move utilization to normal operating levels. To set the parameter to automatic, set the new value to -1. To increase the value, set the new value of DBHEAP to be equal to (pool_cur_size / (4096*U)) where U is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then U = 0.6 * 0.75 = 0.45 (or 45%).

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.

3. On the Performance tab, update the database heap size parameter as recommended above and click OK to apply the update.

From the Command Line Processor, type as shown in the following example:

```
CONNECT TO DATABASE database
UPDATE DATABASE CONFIGURATION FOR database USING DBHEAP size
CONNECT RESET
```

Related reference:

- "Health Indicators" on page 485
- "Monitor Heap Utilization - db2.mon_heap_utilization" on page 521
- "Query Heap Utilization - db2.query_heap_utilization" on page 522
- "Utility Heap Utilization - db.utility_heap_utilization" on page 525
- "Application Control Heap Utilization - db.applctl_heap_utilization" on page 526
- "Application Heap Utilization - db.appl_heap_utilization" on page 528

Utility Heap Utilization - db.utility_heap_utilization

Description:

- This indicator tracks the consumption of the utility heap memory, based on the memory pool with the ID SQLM_HEAP_UTILITY.
- The utilization is calculated using the formula

  ```
  (db.pool_cur_size / db.pool_max_size) * 100
  ```

  for the Memory Pool Identifier SQLM_HEAP_UTILITY.
- Once this percentage reaches the maximum, 100%, utility operations may be impacted.

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and effect changes directly to configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
  3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance. Utility Heap is listed as Backup/Restore/Utility Heap

under the Database Manager Memory group for each instance and database. On Windows, it is listed under the Database Manager Shared Memory group.

4. Click on the check box on the Show Plot column for the Backup/Restore/Utility Heap row to add the element to the plot.

- Increase the utility heap size

  Increase the database configuration parameter UTIL_HEAP_SZ sufficiently to move utilization to normal operating levels. Set the new value of UTIL_HEAP_SZ to be equal to (pool_cur_size / (4096*U)) where U is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then U = 0.6 * 0.75 = 0.45 (or 45%).

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Performance tab, update the utilities heap size parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

  ```
  CONNECT TO DATABASE database
  UPDATE DATABASE CONFIGURATION FOR database USING UTIL_HEAP_SZ size
  CONNECT RESET
  ```

Related reference:

- "Health Indicators" on page 485
- "Monitor Heap Utilization - db2.mon_heap_utilization" on page 521
- "Query Heap Utilization - db2.query_heap_utilization" on page 522
- "Database Heap Utilization - db.database_heap_utilization" on page 524
- "Application Control Heap Utilization - db.applctl_heap_utilization" on page 526
- "Application Heap Utilization - db.appl_heap_utilization" on page 528

Application Control Heap Utilization - db.applctl_heap_utilization

Description:

- This indicator tracks the consumption of the application control heap memory, based on the memory pool with the ID SQLM_HEAP_APPL_CONTROL.
- The utilization is calculated using the formula

  ```
  (db.pool_cur_size / db.pool_max_size) * 100
  ```

for the Memory Pool Identifier SQLM_HEAP_APPL_CONTROL.

- Once this percentage reaches the maximum, 100%, application operations may fail.

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and effect changes directly to configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
  3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance. Application Control Heap is listed under the Agent Private Memory group for each agent.
  4. Click on the check box on the Show Plot column for the Application Control Heap row to add the element to the plot.

- Increase the application control heap size

  Increase the database configuration parameter APP_CTL_HEAP_SZ sufficiently to move utilization to normal operating levels. Set the new value of APP_CTL_HEAP_SZ to be equal to (pool_cur_size / (4096*U)) where U is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then U = 0.6 * 0.75 = 0.45 (or 45%).

  From the Control Center:

  1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
  2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
  3. On the Performance tab, update the maximum application control heap size parameter as recommended above and click OK to apply the update.

  From the Command Line Processor, type as shown in the following example:

```
CONNECT TO DATABASE database
UPDATE DATABASE CONFIGURATION FOR database USING APP_CTL_HEAP_SZ size
CONNECT RESET
```

Related reference:

Application Heap Utilization - db.appl_heap_utilization

Description:

- This indicator tracks the consumption of the application heap memory, based on the memory pool with the ID SQLM_HEAP_APPLICATION.
- The utilization is calculated using the formula

  ```
  (db.pool_cur_size / db.pool_max_size) * 100
  ```

  for the Memory Pool Identifier SQLM_HEAP_APPLICATION.
- Once this percentage reaches the maximum, 100%, applications may fail.

Resolution:

- Investigate memory usage

  Use the Memory Visualizer to view a complete picture of memory usage in DB2. You will be able to monitor overall memory usage and effect changes directly to configuration parameters.

  From the Control Center:

  1. From the Control Center, expand the object tree until you find the Instances folder, and expand the folder until you find the instance that you want.
  2. Right-click the instance, and click View Memory Usage from the pop-up menu. The Memory Visualizer opens.
  3. The Memory Visualizer displays a hierarchical list of all memory pools for the instance, database, and agent. Application Heap is listed under the Agent Private Memory group for each agent.
  4. Click on the check box on the Show Plot column for the Application Heap row to add the element to the plot.
- Increase the application heap size

  Increase the database configuration parameter APPLHEAPSZ sufficiently to move utilization to normal operating levels. Set the new value of APPLHEAPSZ to be equal to (pool_cur_size / (4096*U)) where U is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then U = 0.6 * 0.75 = 0.45 (or 45%).

  From the Control Center:

1. Expand the object tree until you find the Databases folder and expand the folder until you find the database that you want.
2. Right-click the database, and click Configure... from the pop-up menu. The Configure Database dialog opens.
3. On the Performance tab, update the application heap size parameter as recommended above and click OK to apply the update.

From the Command Line Processor, type as shown in the following example:

```
CONNECT TO DATABASE database
UPDATE DATABASE CONFIGURATION FOR database USING APPLHEAPSZ size
CONNECT RESET
```

Related reference:

- "Health Indicators" on page 485
- "Monitor Heap Utilization - db2.mon_heap_utilization" on page 521
- "Query Heap Utilization - db2.query_heap_utilization" on page 522
- "Database Heap Utilization - db.database_heap_utilization" on page 524
- "Utility Heap Utilization - db.utility_heap_utilization" on page 525
- "Application Control Heap Utilization - db.applctl_heap_utilization" on page 526

Appendix B. Version 5 Monitor Output

Version 5 system monitor output

In DB2® Version 6, the self-describing data stream became the standard form of system monitor output to files and pipes. Previously, system monitor data was returned in fixed data structures.

Event monitors write their data in the self-describing monitor format by default. This can be overridden for individual event monitors by setting the registry variable *DB2OLDEVMON=evmon1,evmon2,...*, where *evmon1* is an event monitor that will write its data in the pre-Version 6 format.

When a Version 8 snapshot request is made, but a pre-Version 6 set of snapshot data is returned from the server (for example, from a down-level server), SQLCODE +1627W is returned to the caller. The monitor output is in the pre-Version 6 format and must be parsed using the Version 5 method.

For the opposite situation, where a pre-Version 6 snapshot request is issued, and a Version 8 set of snapshot data is returned, you can use the db2ConvMonStream API. This API can be used to convert the new monitor format for a logical data grouping to the corresponding pre-Version 6 data structure.

The following table lists the snapshot scenarios available with Version 8 clients.

Table 657. Client/server snapshot scenarios

Snapshot Version Requested	Server Version	Data Format Returned	Action
SQLM_DBMON_VERSION1 SQLM_DBMON_VERSION2 SQLM_DBMON_VERSION5 SQLM_DBMON_VERSION5_2	DB2 Version 6 through Version 8	fixed size structures	Parse the data stream using the fixed structure method.
SQLM_DBMON_VERSION6	DB2 Version 6, Version 7, and Version 8	self-describing	The db2ConvMonStream() API can be used to make migrating pre-Version 6 monitor applications easier.
SQLM_DBMON_VERSION7 SQLM_DBMON_VERSION8	DB2 Version 6, Version 7, and Version 8	self-describing	The db2ConvMonStream() API can be used to make migrating pre-Version 6 monitor applications easier.

Related concepts:

- “Event monitors” on page 45
- “Snapshot monitor” on page 21

Related reference:

- “db2ConvMonStream - Convert Monitor Stream” in the *Administrative API Reference*

Appendix C. DB2 Universal Database technical information

Overview of DB2 Universal Database technical information

DB2 Universal Database technical information can be obtained in the following formats:

- Books (PDF and hard-copy formats)
- A topic tree (HTML format)
- Help for DB2 tools (HTML format)
- Sample programs (HTML format)
- Command line help
- Tutorials

This section is an overview of the technical information that is provided and how you can access it.

FixPaks for DB2 documentation

IBM may periodically make documentation FixPaks available. Documentation FixPaks allow you to update the information that you installed from the *DB2 HTML Documentation CD* as new information becomes available.

Note: If you do install documentation FixPaks, your HTML documentation will contain more recent information than either the DB2 printed or online PDF manuals.

Categories of DB2 technical information

The DB2 technical information is categorized by the following headings:

- Core DB2 information
- Administration information
- Application development information
- Business intelligence information
- DB2 Connect information
- Getting started information
- Tutorial information
- Optional component information
- Release notes

The following tables describe, for each book in the DB2 library, the information needed to order the hard copy, print or view the PDF, or locate the HTML directory for that book. A full description of each of the books in

the DB2 library is available from the IBM Publications Center at www.ibm.com/shop/publications/order

The installation directory for the HTML documentation CD differs for each category of information:

htmlcdpath/doc/htmlcd/*%L*/*category*

where:

- *htmlcdpath* is the directory where the HTML CD is installed.
- *%L* is the language identifier. For example, en_US.
- *category* is the category identifier. For example, core for the core DB2 information.

In the PDF file name column in the following tables, the character in the sixth position of the file name indicates the language version of a book. For example, the file name db2d1e80 identifies the English version of the *Administration Guide: Planning* and the file name db2d1g80 identifies the German version of the same book. The following letters are used in the sixth position of the file name to indicate the language version:

Language	Identifier
Arabic	w
Brazilian Portuguese	b
Bulgarian	u
Croatian	9
Czech	x
Danish	d
Dutch	q
English	e
Finnish	y
French	f
German	g
Greek	a
Hungarian	h
Italian	i
Japanese	j
Korean	k
Norwegian	n
Polish	p
Portuguese	v
Romanian	8
Russian	r
Simp. Chinese	c
Slovakian	7
Slovenian	l

Spanish	z
Swedish	s
Trad. Chinese	t
Turkish	m

No form number indicates that the book is only available online and does not have a printed version.

Core DB2 information

The information in this category covers DB2 topics that are fundamental to all DB2 users. You will find the information in this category useful whether you are a programmer, a database administrator, or you work with DB2 Connect, DB2 Warehouse Manager, or other DB2 products.

The installation directory for this category is doc/htmlcd/%L/core.

Table 658. Core DB2 information

Name	Form Number	PDF File Name
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x80
IBM DB2 Universal Database Glossary	No form number	db2t0x80
IBM DB2 Universal Database Master Index	SC09-4839	db2w0x80
IBM DB2 Universal Database Message Reference, Volume 1	GC09-4840	db2m1x80
IBM DB2 Universal Database Message Reference, Volume 2	GC09-4841	db2m2x80
IBM DB2 Universal Database What's New	SC09-4848	db2q0x80

Administration information

The information in this category covers those topics required to effectively design, implement, and maintain DB2 databases, data warehouses, and federated systems.

The installation directory for this category is doc/htmlcd/%L/admin.

Table 659. Administration information

Name	Form number	PDF file name
IBM DB2 Universal Database Administration Guide: Planning	SC09-4822	db2d1x80
IBM DB2 Universal Database Administration Guide: Implementation	SC09-4820	db2d2x80
IBM DB2 Universal Database Administration Guide: Performance	SC09-4821	db2d3x80
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x80
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx80
IBM DB2 Universal Database Data Recovery and High Availability Guide and Reference	SC09-4831	db2hax80
IBM DB2 Universal Database Data Warehouse Center Administration Guide	SC27-1123	db2ddx80
IBM DB2 Universal Database Federated Systems Guide	GC27-1224	db2fpx80
IBM DB2 Universal Database Guide to GUI Tools for Administration and Development	SC09-4851	db2atx80
IBM DB2 Universal Database Replication Guide and Reference	SC27-1121	db2e0x80
IBM DB2 Installing and Administering a Satellite Environment	GC09-4823	db2dsx80
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x80
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x80
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x80

Application development information

The information in this category is of special interest to application developers or programmers working with DB2. You will find information about supported languages and compilers, as well as the documentation required to access DB2 using the various supported programming interfaces, such as embedded SQL, ODBC, JDBC, SQLj, and CLI. If you view this information online in HTML you can also access a set of DB2 sample programs in HTML.

The installation directory for this category is doc/htmlcd/%L/ad.

Table 660. Application development information

Name	Form number	PDF file name
IBM DB2 Universal Database Application Development Guide: Building and Running Applications	SC09-4825	db2axx80
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x80
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x80
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx80
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx80

Business intelligence information

The information in this category describes how to use components that enhance the data warehousing and analytical capabilities of DB2 Universal Database.

The installation directory for this category is doc/htmlcd/%L/wareh.

Table 661. Business intelligence information

Name	Form number	PDF file name
IBM DB2 Warehouse Manager Information Catalog Center Administration Guide	SC27-1125	db2dix80
IBM DB2 Warehouse Manager Installation Guide	GC27-1122	db2idx80

DB2 Connect information

The information in this category describes how to access host or iSeries data using DB2 Connect Enterprise Edition or DB2 Connect Personal Edition.

The installation directory for this category is doc/htmlcd/%L/conn.

Table 662. DB2 Connect information

Name	Form number	PDF file name
APPC, CPI-C, and SNA Sense Codes	No form number	db2apx80
IBM Connectivity Supplement	No form number	db2h1x80
IBM DB2 Connect Quick Beginnings for DB2 Connect Enterprise Edition	GC09-4833	db2c6x80
IBM DB2 Connect Quick Beginnings for DB2 Connect Personal Edition	GC09-4834	db2c1x80
IBM DB2 Connect User's Guide	SC09-4835	db2c0x80

Getting started information

The information in this category is useful when you are installing and configuring servers, clients, and other DB2 products.

The installation directory for this category is doc/htmlcd/%L/start.

Table 663. Getting started information

Name	Form number	PDF file name
IBM DB2 Universal Database Quick Beginnings for DB2 Clients	GC09-4832	db2itx80

Table 663. Getting started information (continued)

Name	Form number	PDF file name
IBM DB2 Universal Database Quick Beginnings for DB2 Servers	GC09-4836	db2isx80
IBM DB2 Universal Database Quick Beginnings for DB2 Personal Edition	GC09-4838	db2i1x80
IBM DB2 Universal Database Installation and Configuration Supplement	GC09-4837	db2iyx80
IBM DB2 Universal Database Quick Beginnings for DB2 Data Links Manager	GC09-4829	db2z6x80

Tutorial information

Tutorial information introduces DB2 features and teaches how to perform various tasks.

The installation directory for this category is doc/htmlcd/%L/tutr.

Table 664. Tutorial information

Name	Form number	PDF file name
Business Intelligence Tutorial: Introduction to the Data Warehouse	No form number	db2tux80
Business Intelligence Tutorial: Extended Lessons in Data Warehousing	No form number	db2tax80
Development Center Tutorial for Video Online using Microsoft Visual Basic	No form number	db2tdx80
Information Catalog Center Tutorial	No form number	db2aix80
Video Central for e-business Tutorial	No form number	db2twx80
Visual Explain Tutorial	No form number	db2tvx80

Optional component information

The information in this category describes how to work with optional DB2 components.

The installation directory for this category is doc/htmlcd/%L/opt.

Table 665. Optional component information

Name	Form number	PDF file name
IBM DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide	GC27-1235	db2lsx80
IBM DB2 Spatial Extender User's Guide and Reference	SC27-1226	db2sbx80
IBM DB2 Universal Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x80
IBM DB2 Universal Database Net Search Extender Administration and Programming Guide **Note:** HTML for this document is not installed from the HTML documentation CD.	SH12-6740	N/A

Release notes

The release notes provide additional information specific to your product's release and FixPak level. They also provides summaries of the documentation updates incorporated in each release and FixPak.

Table 666. Release notes

Name	Form number	PDF file name
DB2 Release Notes	See note.	See note.
DB2 Installation Notes	Available on product CD-ROM only.	Available on product CD-ROM only.

Note: The HTML version of the release notes is available from the Information Center and on the product CD-ROMs. To view the ASCII file on UNIX-based platforms, see the Release.Notes file. This file is located in the DB2DIR/Readme/%L directory, where %L represents the locale name and DB2DIR represents:

- /usr/opt/db2_08_01 on AIX
- /opt/IBM/db2/V8.1 on all other UNIX operating systems

Related tasks:

- "Printing DB2 books from PDF files" on page 541

- "Ordering printed DB2 books" on page 542
- "Accessing online help" on page 542
- "Finding product information by accessing the DB2 Information Center from the administration tools" on page 546
- "Viewing technical documentation online directly from the DB2 HTML Documentation CD" on page 548

Printing DB2 books from PDF files

You can print DB2 books from the PDF files on the *DB2 PDF Documentation* CD. Using Adobe Acrobat Reader, you can print either the entire book or a specific range of pages.

Prerequisites:

Ensure that you have Adobe Acrobat Reader. It is available from the Adobe Web site at www.adobe.com

Procedure:

To print a DB2 book from a PDF file:

1. Insert the *DB2 PDF Documentation* CD. On UNIX operating systems, mount the DB2 PDF Documentation CD. Refer to your *Quick Beginnings* book for details on how to mount a CD on UNIX operating systems.
2. Start Adobe Acrobat Reader.
3. Open the PDF file from one of the following locations:
   - On Windows operating systems:

     x:\doc*language* directory, where *x* represents the CD-ROM drive letter and *language* represents the two-character territory code that represents your language (for example, EN for English).
   - On UNIX operating systems:

     /cdrom/doc/*%L* directory on the CD-ROM, where */cdrom* represents the mount point of the CD-ROM and *%L* represents the name of the desired locale.

Related tasks:

- "Ordering printed DB2 books" on page 542
- "Finding product information by accessing the DB2 Information Center from the administration tools" on page 546
- "Viewing technical documentation online directly from the DB2 HTML Documentation CD" on page 548

Related reference:

- "Overview of DB2 Universal Database technical information" on page 533

Ordering printed DB2 books

Procedure:

To order printed books:

- Contact your IBM authorized dealer or marketing representative. To find a local IBM representative, check the IBM Worldwide Directory of Contacts at www.ibm.com/planetwide
- Phone `1-800-879-2755` in the United States or `1-800-IBM-4YOU` in Canada.
- Visit the IBM Publications Center at www.ibm.com/shop/publications/order

You can also obtain printed DB2 manuals by ordering Doc Packs for your DB2 product from your IBM Reseller. The Doc Packs are subsets of the manuals in the DB2 library selected to help you to get started using the DB2 product that you purchased. The manuals in the Doc Packs are the same as those that are available in PDF format on the *DB2 PDF Documentation CD* and contain the same content as the documentation that is available on the *DB2 HTML Documentation CD*.

Related tasks:

- "Printing DB2 books from PDF files" on page 541
- "Finding topics by accessing the DB2 Information Center from a browser" on page 544
- "Viewing technical documentation online directly from the DB2 HTML Documentation CD" on page 548

Related reference:

- "Overview of DB2 Universal Database technical information" on page 533

Accessing online help

The online help that comes with all DB2 components is available in three types:

- Window and notebook help
- Command line help
- SQL statement help

Window and notebook help explain the tasks that you can perform in a window or notebook and describe the controls. This help has two types:

- Help accessible from the **Help** button
- Infopops

The **Help** button gives you access to overview and prerequisite information. The infopops describe the controls in the window or notebook. Window and notebook help are available from DB2 centers and components that have user interfaces.

Command line help includes Command help and Message help. Command help explains the syntax of commands in the command line processor. Message help describes the cause of an error message and describes any action you should take in response to the error.

SQL statement help includes SQL help and SQLSTATE help. DB2 returns an SQLSTATE value for conditions that could be the result of an SQL statement. SQLSTATE help explains the syntax of SQL statements (SQL states and class codes).

Note: SQL help is not available for UNIX operating systems.

Procedure:

To access online help:

- For window and notebook help, click **Help** or click that control, then click **F1**. If the **Automatically display infopops** check box on the **General** page of the **Tool Settings** notebook is selected, you can also see the infopop for a particular control by holding the mouse cursor over the control.
- For command line help, open the command line processor and enter:
  - For Command help:

    ```
    ? command
    ```

    where *command* represents a keyword or the entire command.

    For example, `? catalog` displays help for all the CATALOG commands, while `? catalog database` displays help for the CATALOG DATABASE command.
- For Message help:

  ```
  ? XXXnnnnn
  ```

  where *XXXnnnnn* represents a valid message identifier.

  For example, `? SQL30081` displays help about the SQL30081 message.
- For SQL statement 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, while `? 08` displays help for the 08 class code.

Related tasks:

- "Finding topics by accessing the DB2 Information Center from a browser" on page 544
- "Viewing technical documentation online directly from the DB2 HTML Documentation CD" on page 548

Finding topics by accessing the DB2 Information Center from a browser

The DB2 Information Center accessed from a browser enables you to access the information you need to take full advantage of DB2 Universal Database and DB2 Connect. The DB2 Information Center also documents major DB2 features and components including replication, data warehousing, metadata, and DB2 extenders.

The DB2 Information Center accessed from a browser is composed of the following major elements:

Navigation tree
The navigation tree is located in the left frame of the browser window. The tree expands and collapses to show and hide topics, the glossary, and the master index in the DB2 Information Center.

Navigation toolbar
The navigation toolbar is located in the top right frame of the browser window. The navigation toolbar contains buttons that enable you to search the DB2 Information Center, hide the navigation tree, and find the currently displayed topic in the navigation tree.

Content frame
The content frame is located in the bottom right frame of the browser window. The content frame displays topics from the DB2 Information Center when you click on a link in the navigation tree, click on a search result, or follow a link from another topic or from the master index.

Prerequisites:

To access the DB2 Information Center from a browser, you must use one of the following browsers:

- Microsoft Explorer, version 5 or later

- Netscape Navigator, version 6.1 or later

Restrictions:

The DB2 Information Center contains only those sets of topics that you chose to install from the *DB2 HTML Documentation CD*. If your Web browser returns a `File not found` error when you try to follow a link to a topic, you must install one or more additional sets of topics from the *DB2 HTML Documentation CD*.

Procedure:

To find a topic by searching with keywords:

1. In the navigation toolbar, click **Search**.
2. In the top text entry field of the Search window, enter one or more terms related to your area of interest and click **Search**. A list of topics ranked by accuracy displays in the **Results** field. The numerical ranking beside the hit provides an indication of the strength of the match (bigger numbers indicate stronger matches).

   Entering more terms increases the precision of your query while reducing the number of topics returned from your query.
3. In the **Results** field, click the title of the topic you want to read. The topic displays in the content frame.

To find a topic in the navigation tree:

1. In the navigation tree, click the book icon of the category of topics related to your area of interest. A list of subcategories displays underneath the icon.
2. Continue to click the book icons until you find the category containing the topics in which you are interested. Categories that link to topics display the category title as an underscored link when you move the cursor over the category title. The navigation tree identifies topics with a page icon.
3. Click the topic link. The topic displays in the content frame.

To find a topic or term in the master index:

1. In the navigation tree, click the "Index" category. The category expands to display a list of links arranged in alphabetical order in the navigation tree.
2. In the navigation tree, click the link corresponding to the first character of the term relating to the topic in which you are interested. A list of terms with that initial character displays in the content frame. Terms that have multiple index entries are identified by a book icon.

3. Click the book icon corresponding to the term in which you are interested. A list of subterms and topics displays below the term you clicked. Topics are identified by page icons with an underscored title.
4. Click on the title of the topic that meets your needs. The topic displays in the content frame.

Related concepts:

- "Accessibility" on page 553
- "DB2 Information Center accessed from a browser" on page 556

Related tasks:

- "Finding product information by accessing the DB2 Information Center from the administration tools" on page 546
- "Updating the HTML documentation installed on your machine" on page 548
- "Troubleshooting DB2 documentation search with Netscape 4.x" on page 551
- "Searching the DB2 documentation" on page 552

Related reference:

- "Overview of DB2 Universal Database technical information" on page 533

Finding product information by accessing the DB2 Information Center from the administration tools

The DB2 Information Center provides quick access to DB2 product information and is available on all operating systems for which the DB2 administration tools are available.

The DB2 Information Center accessed from the tools provides six types of information.

Tasks Key tasks you can perform using DB2.

Concepts
Key concepts for DB2.

Reference
DB2 reference information, such as keywords, commands, and APIs.

Troubleshooting
Error messages and information to help you with common DB2 problems.

Samples
Links to HTML listings of the sample programs provided with DB2.

Tutorials

Instructional aid designed to help you learn a DB2 feature.

Prerequisites:

Some links in the DB2 Information Center point to Web sites on the Internet. To display the content for these links, you will first have to connect to the Internet.

Procedure:

To find product information by accessing the DB2 Information Center from the tools:

1. Start the DB2 Information Center in one of the following ways:
   - From the graphical administration tools, click on the **Information Center** icon in the toolbar. You can also select it from the **Help** menu.
   - At the command line, enter **db2ic**.
2. Click the tab of the information type related to the information you are attempting to find.
3. Navigate through the tree and click on the topic in which you are interested. The Information Center will then launch a Web browser to display the information.
4. To find information without browsing the lists, click the **Search** icon to the right of the list.

   Once the Information Center has launched a browser to display the information, you can perform a full-text search by clicking the **Search** icon in the navigation toolbar.

Related concepts:

- "Accessibility" on page 553
- "DB2 Information Center accessed from a browser" on page 556

Related tasks:

- "Finding topics by accessing the DB2 Information Center from a browser" on page 544
- "Searching the DB2 documentation" on page 552

Viewing technical documentation online directly from the DB2 HTML Documentation CD

All of the HTML topics that you can install from the *DB2 HTML Documentation CD* can also be read directly from the CD. Therefore, you can view the documentation without having to install it.

Restrictions:

As the Tools help is installed from the DB2 product CD and not from the *DB2 HTML Documentation CD*, you must install the DB2 product to view the help.

Procedure:

1. Insert the *DB2 HTML Documentation* CD. On UNIX operating systems, mount the *DB2 HTML Documentation CD*. Refer to your *Quick Beginnings* book for details on how to mount a CD on UNIX operating systems.
2. Start your HTML browser and open the appropriate file:
   - For Windows operating systems:

     ```
     e:\program files\IBM\SQLLIB\doc\htmlcd\%L\index.htm
     ```

     where *e* represents the CD-ROM drive, and %L is the locale of the documentation that you wish to use, for example, **en_US** for English.
   - For UNIX operating systems:

     ```
     /cdrom/program files/IBM/SQLLIB/doc/htmlcd/%L/index.htm
     ```

     where */cdrom/* represents where the CD is mounted, and %L is the locale of the documentation that you wish to use, for example, **en_US** for English.

Related tasks:

- "Finding topics by accessing the DB2 Information Center from a browser" on page 544
- "Copying files from the DB2 HTML Documentation CD to a Web server" on page 550

Related reference:

- "Overview of DB2 Universal Database technical information" on page 533

Updating the HTML documentation installed on your machine

It is now possible to update the HTML installed from the *DB2 HTML Documentation CD* when updates are made available from IBM. This can be done in one of two ways:

- Using the Information Center (if you have the DB2 administration GUI tools installed).
- By downloading and applying a DB2 HTML documentation FixPak .

Note: This will NOT update the DB2 code; it will only update the HTML documentation installed from the *DB2 HTML Documentation CD*.

Procedure:

To use the Information Center to update your local documentation:

1. Start the DB2 Information Center in one of the following ways:
   - From the graphical administration tools, click on the **Information Center** icon in the toolbar. You can also select it from the **Help** menu.
   - At the command line, enter **db2ic**.
2. Ensure your machine has access to the external Internet; the updater will download the latest documentation FixPak from the IBM server if required.
3. Select **Information Center —> Update Local Documentation** from the menu to start the update.
4. Supply your proxy information (if required) to connect to the external Internet.

The Information Center will download and apply the latest documentation FixPak, if one is available.

To manually download and apply the documentation FixPak :

1. Ensure your machine is connected to the Internet.
2. Open the DB2 support page in your Web browser at: www.ibm.com/software/data/db2/udb/winos2unix/support.
3. Follow the link for Version 8 and look for the ″Documentation FixPaks″ link.
4. Determine if the version of your local documentation is out of date by comparing the documentation FixPak level to the documentation level you have installed. This current documentation on your machine is at the following level: **DB2 v8.1 GA**.
5. If there is a more recent version of the documentation available then download the FixPak applicable to your operating system. There is one FixPak for all Windows platforms, and one FixPak for all UNIX platforms.
6. Apply the FixPak:
   - For Windows operating systems: The documentation FixPak is a self extracting zip file. Place the downloaded documentation FixPak in an empty directory, and run it. It will create a **setup** command which you can run to install the documentation FixPak.

- For UNIX operating systems: The documentation FixPak is a compressed tar.Z file. Uncompress and untar the file. It will create a directory named `delta_install` with a script called **installdocfix**. Run this script to install the documentation FixPak.

Related tasks:

- "Copying files from the DB2 HTML Documentation CD to a Web server" on page 550

Related reference:

- "Overview of DB2 Universal Database technical information" on page 533

Copying files from the DB2 HTML Documentation CD to a Web server

The entire DB2 information library is delivered to you on the *DB2 HTML Documentation CD* and may be installed on a Web server for easier access. Simply copy to your Web server the documentation for the languages that you want.

Note: You might encounter slow performance if you access the HTML documentation from a Web server through a low-speed connection.

Procedure:

To copy files from the *DB2 HTML Documentation CD* to a Web server, use the appropriate source path:

- For Windows operating systems:

  E:\program files\IBM\SQLLIB\doc\htmlcd*%L**.*

  where *E* represents the CD-ROM drive and *%L* represents the language identifier.
- For UNIX operating systems:

  /*cdrom*/program files/IBM/SQLLIB/doc/htmlcd/*%L*/*.*

  where *cdrom* represents the mount point for the CD-ROM drive and *%L* represents the language identifier.

Related tasks:

- "Searching the DB2 documentation" on page 552

Related reference:

- "Supported DB2 interface languages, locales, and code pages" in the *Quick Beginnings for DB2 Servers*
- "Overview of DB2 Universal Database technical information" on page 533

Troubleshooting DB2 documentation search with Netscape 4.x

Most search problems are related to the Java support provided by web browsers. This task describes possible workarounds.

Procedure:

A common problem with Netscape 4.x involves a missing or misplaced security class. Try the following workaround, especially if you see the following line in the browser Java console:

```
Cannot find class  java/security/InvalidParameterException
```

- On Windows operating systems:

  From the *DB2 HTML Documentation CD*, copy the supplied *x*:program files\IBM\SQLLIB\doc\htmlcd*locale*\InvalidParameterException.class file to the java\classes\java\security\ directory relative to your Netscape browser installation, where *x* represents the CD-ROM drive letter and *locale* represents the name of the desired locale.

  Note: You may have to create the java\security\ subdirectory structure.

- On UNIX operating systems:

  From the *DB2 HTML Documentation CD*, copy the supplied */cdrom*/program files/IBM/SQLLIB/doc/htmlcd/*locale*/InvalidParameterException.class file to the java/classes/java/security/ directory relative to your Netscape browser installation, where *cdrom* represents the mount point of the CD-ROM and *locale* represents the name of the desired locale.

  Note: You may have to create the java/security/ subdirectory structure.

If your Netscape browser still fails to display the search input window, try the following:

- Stop all instances of Netscape browsers to ensure that there is no Netscape code running on the machine. Then open a new instance of the Netscape browser and try to start the search again.
- Purge the browser's cache.
- Try a different version of Netscape, or a different browser.

Related tasks:

- "Searching the DB2 documentation" on page 552

Searching the DB2 documentation

You can search the library of DB2 documentation to locate information that you need. A pop-up search window opens when you click the search icon in the navigation toolbar of the DB2 Information Center (accessed from a browser). The search can take a minute to load, depending on the speed of your computer and network.

Prerequisites:

You need Netscape 6.1 or higher, or Microsoft's Internet Explorer 5 or higher. Ensure that your browser's Java support is enabled.

Restrictions:

The following restrictions apply when you use the documentation search:

- Search is not case sensitive.
- Boolean searches are not supported.
- Wildcard and partial searches are not supported. A search on *java** (or *java*) will only look for the literal string *java** (or *java*) and would not, for example, find *javadoc*.

Procedure:

To search the DB2 documentation:

1. In the navigation toolbar, click the **Search** icon.
2. In the top text entry field of the Search window, enter one or more terms (separated by a space) related to your area of interest and click **Search**. A list of topics ranked by accuracy displays in the **Results** field. The numerical ranking beside the hit provides an indication of the strength of the match (bigger numbers indicate stronger matches).

   Entering more terms increases the precision of your query while reducing the number of topics returned from your query.
3. In the **Results** list, click the title of the topic you want to read. The topic displays in the content frame of the DB2 Information Center.

Note: When you perform a search, the first (highest-ranking) result is automatically loaded into your browser frame. To view the contents of other search results, click on the result in the results list.

Related tasks:

- "Troubleshooting DB2 documentation search with Netscape 4.x" on page 551

Online DB2 troubleshooting information

With the release of DB2® UDB Version 8, there will no longer be a *Troubleshooting Guide*. The troubleshooting information once contained in this guide has been integrated into the DB2 publications. By doing this, we are able to deliver the most up-to-date information possible. To find information on the troubleshooting utilities and functions of DB2, access the DB2 Information Center from any of the tools.

Refer to the DB2 Online Support site if you are experiencing problems and want help finding possible causes and solutions. The support site contains a large, constantly updated database of DB2 publications, TechNotes, APAR (product problem) records, FixPaks, and other resources. You can use the support site to search through this knowledge base and find possible solutions to your problems.

Access the Online Support site at www.ibm.com/software/data/db2/udb/winos2unix/support, or by clicking the **Online Support** button in the DB2 Information Center. Frequently changing information, such as the listing of internal DB2 error codes, is now also available from this site.

Related concepts:

- "DB2 Information Center accessed from a browser" on page 556

Related tasks:

- "Finding product information by accessing the DB2 Information Center from the administration tools" on page 546

Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully. These are the major accessibility features in DB2® Universal Database Version 8:

- DB2 allows you to operate all features using the keyboard instead of the mouse. See "Keyboard Input and Navigation" on page 554.
- DB2 enables you customize the size and color of your fonts. See "Accessible Display" on page 554.
- DB2 allows you to receive either visual or audio alert cues. See "Alternative Alert Cues" on page 554.
- DB2 supports accessibility applications that use the Java™ Accessibility API. See "Compatibility with Assistive Technologies" on page 554.

- DB2 comes with documentation that is provided in an accessible format. See "Accessible Documentation".

Keyboard Input and Navigation

Keyboard Input

You can operate the DB2 Tools using only the keyboard. You can use keys or key combinations to perform most operations that can also be done using a mouse.

Keyboard Focus

In UNIX-based systems, the position of the keyboard focus is highlighted, indicating which area of the window is active and where your keystrokes will have an effect.

Accessible Display

The DB2 Tools have features that enhance the user interface and improve accessibility for users with low vision. These accessibility enhancements include support for customizable font properties.

Font Settings

The DB2 Tools allow you to select the color, size, and font for the text in menus and dialog windows, using the Tools Settings notebook.

Non-dependence on Color

You do not need to distinguish between colors in order to use any of the functions in this product.

Alternative Alert Cues

You can specify whether you want to receive alerts through audio or visual cues, using the Tools Settings notebook.

Compatibility with Assistive Technologies

The DB2 Tools interface supports the Java Accessibility API enabling use by screen readers and other assistive technologies used by people with disabilities.

Accessible Documentation

Documentation for the DB2 family of products is available in HTML format. This allows you to view documentation according to the display preferences set in your browser. It also allows you to use screen readers and other assistive technologies.

DB2 tutorials

The DB2® tutorials help you learn about various aspects of DB2 Universal Database. The tutorials provide lessons with step-by-step instructions in the areas of developing applications, tuning SQL query performance, working with data warehouses, managing metadata, and developing Web services using DB2.

Before you begin:

Before you can access these tutorials using the links below, you must install the tutorials from the *DB2 HTML Documentation* CD.

If you do not want to install the tutorials, you can view the HTML versions of the tutorials directly from the *DB2 HTML Documentation CD*. PDF versions of these tutorials are also available on the *DB2 PDF Documentation CD*.

Some tutorial lessons use sample data or code. See each individual tutorial for a description of any prerequisites for its specific tasks.

DB2 Universal Database tutorials:

If you installed the tutorials from the *DB2 HTML Documentation* CD, you can click on a tutorial title in the following list to view that tutorial.

Business Intelligence Tutorial: Introduction to the Data Warehouse Center
Perform introductory data warehousing tasks using the Data Warehouse Center.

Business Intelligence Tutorial: Extended Lessons in Data Warehousing
Perform advanced data warehousing tasks using the Data Warehouse Center.

Development Center Tutorial for Video Online using Microsoft® Visual Basic
Build various components of an application using the Development Center Add-in for Microsoft Visual Basic.

Information Catalog Center Tutorial
Create and manage an information catalog to locate and use metadata using the Information Catalog Center.

Video Central for e-business Tutorial
Develop and deploy an advanced DB2 Web Services application using WebSphere® products.

Visual Explain Tutorial
Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

DB2 Information Center accessed from a browser

The DB2® Information Center gives you access to all of the information you need to take full advantage of DB2 Universal Database™ and DB2 Connect™ in your business. The DB2 Information Center also documents major DB2 features and components including replication, data warehousing, the Information Catalog Center, Life Sciences Data Connect, and DB2 extenders.

The DB2 Information Center accessed from a browser has the following features if you view it in Netscape Navigator 6.1 or later or Microsoft Internet Explorer 5 or later. Some features require you to enable support for Java or JavaScript:

Regularly updated documentation
Keep your topics up-to-date by downloading updated HTML.

Search
Search all of the topics installed on your workstation by clicking **Search** in the navigation toolbar.

Integrated navigation tree
Locate any topic in the DB2 library from a single navigation tree. The navigation tree is organized by information type as follows:

- Tasks provide step-by-step instructions on how to complete a goal.
- Concepts provide an overview of a subject.
- Reference topics provide detailed information about a subject, including statement and command syntax, message help, requirements.

Master index
Access the information installed from the *DB2 HTML Documentation CD* from the master index. The index is organized in alphabetical order by index term.

Master glossary
The master glossary defines terms used in the DB2 Information Center. The glossary is organized in alphabetical order by glossary term.

Related tasks:

- "Finding topics by accessing the DB2 Information Center from a browser" on page 544
- "Finding product information by accessing the DB2 Information Center from the administration tools" on page 546
- "Updating the HTML documentation installed on your machine" on page 548

Appendix D. 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

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both, and have been used in at least one of the documents in the DB2 UDB documentation library.

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

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 UDB documentation library:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium 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.

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

Index

E

F

G

H

I

L

M

N

O

P

T

U

V

W

X

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-237-5511 for customer service
- 1-888-426-4343 to learn about available service options
- 1-800-IBM-4YOU (426-4968) for DB2 marketing and sales

In Canada, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) for customer service
- 1-800-465-9600 to learn about available service options
- 1-800-IBM-4YOU (1-800-426-4968) for DB2 marketing and sales

To locate an IBM office in your country or region, check IBM's Directory of Worldwide Contacts on the web at www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, client downloads, newsgroups, FixPaks, news, and links to web resources.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) to order products or to obtain general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, go to the IBM Worldwide page at www.ibm.com/planetwide

Printed in U.S.A.

SC09-4847-00

Spine information:

IBM® DB2 Universal Database™ System Monitor Guide and Reference Version 8