IBM® DB2® Universal Database

Troubleshooting Guide

GC09-2850-01

IBM® DB2® Universal Database

Troubleshooting Guide

GC09-2850-01

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).

Contents

Chapter 1. Troubleshooting overview

7 Introduction to problem determination

7 The first step in good problem analysis is to describe the problem completely.
7 Without a problem description, you will not know where to start investigating the
7 cause of the problem. This step includes asking yourself such basic questions as:

- 7 What are the symptoms?
- 7 Where is the problem happening?
- 7 When does the problem happen?
- 7 Under which conditions does the problem happen?
- 7 Is the problem reproducible?

7 Answering these and other questions will lead to a good description to most
7 problems, and is the best way to start down the path of problem resolution.

7 **What is the problem?:**

7 When starting to describe a problem, the most obvious question is ″What is the
7 problem?″ This may seem like a straightforward question; however it can be
7 broken down into several other questions to create a more descriptive picture of
7 the problem. These questions can include:

- 7 Who or what is reporting the problem?
- 7 What are the error codes and error messages?
- 7 What are the symptoms?
- 7 How does it fail? For example: loop, hang, crash, performance degradation,
  7 incorrect result.
- 7 Can the problem be reproduced?
- 7 What is the business impact?

7 The following problem scenario demonstrates how and when you might ask these
7 questions.

7 **What is the environment and configuration?:**

7 Determining where the problem originates is not always easy, but is one of the
7 most important steps in resolving a problem. Many layers of technology almost
7 always exist between the reporting and failing components. Networks, disks, and
7 drivers are only a few components to be considered when you are investigating
7 problems.

- 7 Is the problem platform specific, or common to multiple platforms?
- 7 Is the current enviroment and configuration supported?
- 7 Is the application running local on the database server or on a remote server?
- 7 Is there a gateway involved?
- 7 Does the database reside on individual disks, or on a RAID array?

7 These types of questions will help you isolate the problem layer, and are necessary
7 to determine the problem source. Remember that just because one layer is
7 reporting a problem, it does not always mean the root cause exists there.

7 Part of identifying where a problem is occurring is understanding the environment
7 in which it exists. You should always take some time to completely describe the
7 problem environment, including the operating system, its version, all
7 corresponding software and versions, and hardware information. Confirm you are
7 running within an environment that is a supported configuration, as many
7 problems can be explained by discovering software levels that are not meant to run
7 together, or have not been fully tested together.

7 **When does the problem happen?:**

7 Developing a detailed time line of events leading up to a failure is another
7 necessary step in problem analysis, especially for those cases that are one-time
7 occurrences. You can most easily do this by working backwards --start at the time
7 an error was reported (as exact as possible, even down to milliseconds), and work
7 backwards through available logs and information. Usually you only have to look
7 as far as the first suspicious event that you find in any diagnostic log, however this
7 is not always easy to do and will only come with practice. Knowing when to stop
7 is especially difficult when there are multiple layers of technology each with its
7 own diagnostic information.

7 - Does the problem only happen at a certain time of day or night?
7 - How often does this happen?
7 - What sequence of events lead up to the time the problem is reported?
7 - Does the problem happen after an environment change such as upgrading
7 existing or installing new software or hardware?

7 Responding to questions like this will help you create a detailed time line of
7 events, and will provide you with a frame of reference in which to investigate.

7 **Under which conditions does the problem happen?:**

7 Knowing what else is running at the time of a problem is important for any
7 complete problem description. If a problem occurs in a certain environment or
7 under certain conditions, that can be a key indicator of the problem cause.

7 - Does the problem always occur when performing the same task?
7 - Does a certain sequence of events need to occur for the problem to surface?
7 - Do other applications fail at the same time?

7 Answering these types of questions will help you explain the environment in
7 which the problem occurs, and correlate any dependencies. Remember that just
7 because multiple problems may have occurred around the same time, it does not
7 necessarily mean that they are always related.

7 **Is the problem reproducible?:**

7 From a problem description and investigation standpoint, the ″ideal″ problem is
7 one that is reproducible. With reproducible problems you almost always have a
7 larger set of tools or procedures available to use to help your investigation.
7 Consequently reproducible problems are usually easier to debug and solve.

7 However, reproducible problems can have a disadvantage: if the problem is of
7 significant business impact, you don’t want it recurring. If possible, recreating the
7 problem in a test or development environment is often preferable in this case.

7 • Can the problem be recreated on a test machine?
7 • Are multiple users or applications encountering the same type of problem?
7 • Can the problem be recreated by running a single command, a set of commands,
7 or a particular application, or a standalone application?
7 • Can the problem be recreated by entering the equivalent command/query from
7 a DB2 command line?

7 Recreating a single incident problem in a test or development environment is often
7 preferable, as there is usually much more flexibility and control when
7 investigating.

7 Our example is reproducible.

7 **Conclusion:**

7 Describing a problem accurately and completely may be easy for some problems,
7 but very difficult for others. (The difficulty usually increases as the environmental
7 complexity increases). However, the questions that you need to ask are usually the
7 same: who, what, where, when, and how.

7 The db2support tool poses subsets of the questions described above when you
7 specify the ″-q″ option. This information is then used by DB2 Support to
7 understand the context in which you are encountering your problem. For example:

7
```
db2support . -q
```

7 Once have a good understanding of the problem situation, you can research
7 similar problems, and learn about workarounds, and fixes in the available DB2
7 troubleshooting resources.

7 **Related concepts:**

7 • on page 0

Identifying the version and service level of your product

The db2level command will help you determine the version and service level (build level and FixPak number) of your DB2 instance. To determine if your DB2 instance is at the latest service level, compare your db2level output to the build level and FixPak information on the FixPak download pages at the DB2 UDB support web site.

A typical result of running the db2level command on a Windows system would be:

```
C:\>db2level

DB21085I  Instance "DB2" uses "32" bits and DB2 code release "SQL08020"
with level identifier "03010106".
Informational tokens are "DB2 v8.1.7.664", "s040914", "WR21342", and FixPak "7".
Product is installed at "D:\IBM\SQLLIB".
```

The combination of the four informational tokens uniquely identify the precise service level of your DB2 instance. This information is essntial when contacting

IBM support for assistance. In particular, the second informational token shows the date that the product was built. In the above example, ″s040914″ indicates a build date of September 14th, 2004.

For JDBC or SQLJ application, if you are using the Universal Driver for SQLJ and JDBC, you can determine the level of the driver by running the db2jcc utility:

```
db2jcc -version

IBM DB2 JDBC Universal Driver Architecture 2.3.63
```

On Windows, this command is in the sqllib\bin directory.

On UNIX/Linux, this command is in the sqllib/bin directory.

DB2 server process model

Knowledge of the DB2® process model can help you determine the nature of a problem because it helps you to understand how the database manager and its associated components interact.

The process model used by all DB2 servers facilitates the communication that occurs between database servers and clients and local applications. It also ensures that database applications are isolated from resources such as database control blocks and critical database files.

UNIX-based environments use an architecture based on **system processes**. For example, the DB2 communications listeners are created as system processes. Intel operating systems such as Windows® use an architecture based on **threads** to maximize performance. Unless explicitly noted, this discussion uses the term ″process″ to refer to both processes and threads. You can find details of the differences between the use of Windows threads and UNIX® processes later in this topic.

For each database being accessed, various processes are started to deal with the various database tasks such as prefetching, communication, and logging.

Each process of a client application has a single **coordinator agent** that operates on a database. A coordinator agent works on behalf of an application, and communicates to other agents, using interprocess communication (IPC) or remote communication protocols.

DB2 architecture provides a **firewall** so that applications run in a different address space from DB2. The firewall protects the database and the database manager from applications, stored procedures, and user-defined functions (UDFs). A firewall maintains the integrity of the data in the databases, because it disables application programming errors from overwriting internal buffers or files of the database manager. The firewall also improves reliability, because application errors cannot crash the database manager.

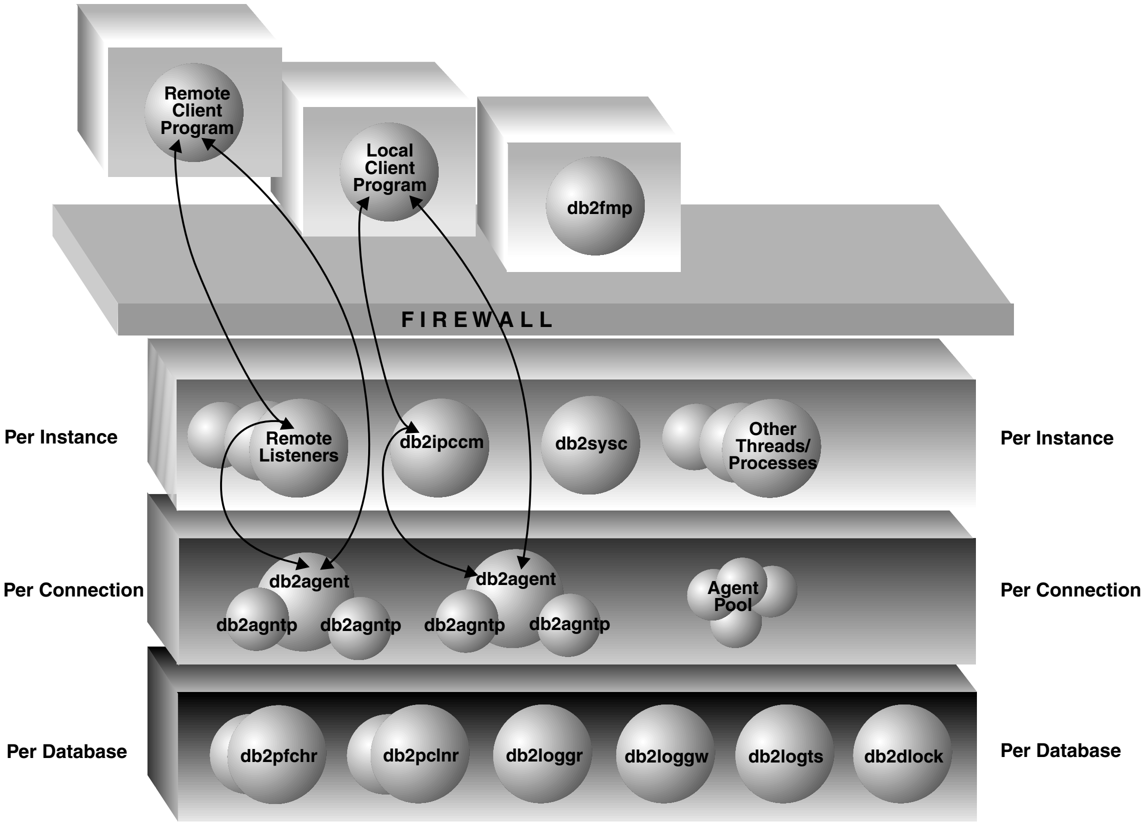

Figure 1. Process Model for DB2 Systems

The following list provides additional details on the processes shown in the figure:

Client Programs:

Client programs run remotely or on the same machine as the database server. They make their first contact with the database through a listener. A coordinator agent (db2agent) is then assigned to them.

Listeners:

Client programs make initial contact with communication listeners, which are started when DB2 is started. There is a listener for each configured communication protocol, and an interprocess communications (IPC) listener (db2ipccm) for local client programs. Listeners include:

- **db2ipccm**, for local client connections
- **db2tcpcm**, for TCP/IP connections
- **db2snacm**, for APPC connections
- **db2tcpdm**, for TCP/IP discovery tool requests

Agents:

All connection requests from client applications, whether they are local or remote, are allocated a corresponding coordinator agent (**db2agent**). When the coordinator agent is created, it performs all database requests on behalf of the application.

In some environments, in which the *intra_parallel* database manager configuration parameter is enabled, the coordinator agent distributes the database requests to subagents (**db2agntp**). These agents perform the requests for the application. Once the coordinator agent is created, it handles all database requests on behalf of its application by coordinating subagents (**db2agntp**) that perform requests on the database.

A coordinator agent may be:

- Connected to the database with an alias. For example, ″db2agent (DATA1)″ is connected to the database alias ″DATA1″.
- Attached to an instance. For example, ″db2agent (user1)″ is attached to the instance ″user1″.

An additional type of agent, **db2agnsc** is created by DB2 to perform certain operations in parallel. In particular, they are used in database recovery actions.

Idle agents reside in an agent pool. These agents are available for requests from coordinator agents operating on behalf of client programs, or from subagents operating on behalf of existing coordinator agents. The number of available agents is dependent on the database manager configuration parameters *maxagents* and *num_poolagents*.

db2fmp:

The fenced mode process. It is responsible for executing fenced stored procedures and user-defined functions outside the firewall. db2fmp is always a separate process but may be multithreaded depending on the types of routines it executes.

Database Threads/Processes:

The following list includes some of the important threads/processes used by each database:

- **db2pfchr**, for buffer pool prefetchers
- **db2pclnr**, for buffer pool page cleaners
- **db2loggr**, for manipulating log files to handle transaction processing and recovery
- **db2loggw**, for writing log records to the log files.
- **db2logts**, for collecting historical information about which logs are active when a tablespace is modified. This information is ultimately recorded in the DB2TSCHG.HIS file in the database directory. It is used to speed up tablespace rollforward recovery.
- **db2dlock**, for deadlock detection. In a multi-partitioned database environment, an additional process called **db2glock** is used to coordinate the information gathered from the db2dlock process on each partition. db2glock runs only on the catalog partition.

Database Server Threads and Processes:

The system controller (**db2sysc**) must exist in order for the database server to function. Also, the following threads and processes may be started to carry out various tasks:

- **db2resyn**, the resync agent that scans the global resync list

- **db2gds**, the global daemon spawner on UNIX-based systems that starts new processes
- **db2wdog**, the watchdog on UNIX-based systems that handles abnormal terminations
- **db2fcmdm**, the fast communications manager daemon for handling inter-partition communication (used only in multi-partitioned databases)
- **db2pdbc**, the parallel system controller, which handles parallel requests from remote nodes (used only in a partitioned database environment).
- **db2cart**, for archiving log files when accessing a database configured with USEREXIT enabled
- **db2fmtlg**, for formatting log files, when accessing a database configured with LOGRETAIN enabled, but with USEREXIT disabled
- **db2panic**, the panic agent, which handles urgent requests after agent limits have been reached at a particular node (used only in a partitioned database environment)
- **dlasync**, a monitor for the Data Links servers, if you have configured DB2 for datalinks

Differences between Windows and UNIX:

DB2 for Windows differs from UNIX-based environments in that the database engine is multi-threaded, not multi-processed. In Windows systems, each of the dispatchable units on the agent side of the firewall is a thread under the process db2sysc. This allows the database engine to let the operating system perform task-switching at the thread level instead of the process level. For each database being accessed, threads are started to deal with database tasks (for example, prefetching).

Another difference is in the handling of abnormal terminations. There is no need for a ″watchdog″ process in Windows systems, because these systems ensure that the allocated resources are cleaned up after an abnormal termination. Thus, there is no equivalent of the db2wdog process on the Windows systems. In addition, the db2gds process or thread is not needed on the Windows systems, which have their own mechanisms for starting threads.

Related concepts:

- on page 0
- “Determining active process status” on page 41

Proactive monitoring tools

Database monitoring is an important part of tuning a database server. You should develop a database monitoring plan. This not only allows for easier detection of a problem, but also enables you to refer back to historical data to isolate a change, extrapolate the effects of a change, better understand user and application activity and behaviors, and, of course, improve database performance!

Monitoring does involve overhead and this should be taken into account when developing the monitoring plan. The frequency and volume of monitoring needs to be controlled.

Techniques for monitoring the database:

DB2 has two main tools that allow you to access system monitor information, each serving a particular purpose:

- Snapshot monitoring - captures a picture of the database status at a particular point in time (the moment the snapshot is taken). Snapshot monitoring is useful in determining the current state of the database and its applications.
- Event monitoring - provides information about specific database events which occur over a period of time. Event monitoring is useful in identifying database or application state changes. It is used to gather details about changes regarding different events such as connections, tables, statements, etc.

In order to allow for effective memory usage, it is possible to configure how much memory is used for database monitoring. The database system monitor heap size cofiguration parameter MON_HEAP_SZ, which is a database manager configuration parameter, specifies the amount of memory allocated on a DB2START and is used when monitoring takes place. Setting this parameter to 0 (zero) will not allow any monitoring to take place.

Note: You must have one of SYSADM, SYSCTRL, SYSMAINT, or SYSMON authority to perform much of the set-up related to monitoring.

Levels of monitoring:

The monitoring utilities collect information at different levels within the system:

- *Database Manager* - the monitor gathers details from the time the instance is started till the time it is stopped.
- *Database* - this level of monitoring begins on the first connection or activation, and will continue until deactivation or the last connection is terminated.
- *Application* - this level of monitoring begins when an application connects to the database and continues until it disconnects from the database.

You can control the type of data to collect. The type of information is classified as follows:

BUFFERPOOL ACTIVITY
LOCKS
SORTS
SQL STATEMENTS
TABLE ACTIVITY
TIMESTAMPS
UNIT OF WORK

So, for example, if you notice long lock waits, it would be logical to monitor the LOCK group in an attempt to isolate the problem.

Snapshot and event monitors

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. What follows are brief overviews to get you started with snapshot monitors and event monitors. For more information, refer to the System Monitor Guide and Reference, or the ″Monitoring″ section of the Information Center.

Setting up snapshot monitoring

The DB2® monitor maintains a running tally of valuable system information. You can get a summary of system status at any time by issuing the GET SNAPSHOT command. The parameters of the GET SNAPSHOT command determine the level of detail, and the type of monitoring information returned.

Some of the statistics returned by this command provide point- in-time information, others provide cumulative information since the last `reset monitor` command was performed. An example of a cumulative counter would be ″`deadlocks detected`″. This holds the number of deadlocks since the last reset monitor command. An example of an event-specific field would be ″`lock list` memory in use″. This field would contain the number of bytes currently used for the lock list and can change from one iteration of a snapshot to the next, as applications connect and disconnect and obtain and release locks respectively.

As an example, try using the **sample** database to take a snapshot. From a CLP session, issue the following commands:

```
DB2 CONNECT TO sample
DB2 RESET MONITOR ALL
DB2 GET SNAPSHOT FOR DATABASE ON sample
DB2 "SELECT * FROM syscat.bufferpools"
DB2 GET SNAPSHOT FOR DATABASE ON sample
```

Creating an event monitor

The Event Monitor facility within DB2 also allows for a great deal of flexibility. The initial step is to first create an event monitor.

As an example, try creating an event monitor for the SAMPLE database:

```
DB2 CONNECT TO sample
DB2 CREATE EVENT MONITOR evmonex1 FOR STATEMENTS WRITE TO FILE
'c:\temp\testfile' MAXFILES 3 MAXFILESIZE 1024 NONBLOCKED APPEND
```

The above command will create an event monitor called evmonex1 to capture all statement events. A maximum of 3 files of 4MB each will be created. The file directory (in this example ″c:\temp\testfile″) does not have to exist at CREATE EVENT MONITOR time. However, if it does not occur by the time an event monitor is activated, an error will occur (i.e. SQL1614N An I/O error occurred when activating an event monitor. Reason code = ″2″. SQLSTATE=58030).

Activating an event monitor

To enable the newly created event monitor to collect information, you must activate it. For example:

```
DB2 SET EVENT MONITOR evmonex1 STATE 1
```

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, for example:

```
DB2 SELECT evmonname, EVENT_MON_STATE(evmonname) FROM syscat.eventmonitors
```

A returned value of 0 indicates that the specified event monitor is inactive, and 1 indicates that it is active.

For more example commands, including how to disable and drop event monitors, see Collecting information about database system events.

Formatting an event monitor

The output from file or pipe event monitors (as opposed to table event monitors) are written out in an unreadable format. In order to make the output readable, you need to run a tool called db2evmon:

```
db2evmon -db <database alias> -evm <event monitor name>
```

As an exercise, try generating and then formatting the output of the evmonex1 event monitor created earlier.

```
db2evmon -db sample -evm evmonex1
```

Try running that command again after generating a statement event, for example by issuing

```
'DB2 "SELECT * FROM EMPLOYEE"'
```

. There will now be much more information, since an event has been triggered.

Basic monitoring commands

The following commands track current monitor settings, enabling functional groups, and ″clear out″ existing monitor data. These commands can be executed from a DB2 CLP session:

To obtain the current state of the functional groups listed earlier, issue the following command:

```
DB2 GET MONITOR SWITCHES
```

If you see that a particular group's ″switch″ is ″on″, you know that this group will be monitored.

Note: Event monitors are only affected by the time and timestamp information switch. All other switch settings effect snapshot monitors only and have no effect on the data collected by event monitors.

To enable or disable the monitor switches, use the command:

```
DB2 UPDATE MONITOR SWITCHES USING <switch name> ON | 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 a specific partition, issue the following command:

```
db2 update monitor switches using <switch name> ON|OFF at dbpartitionnum <partition number>
```

To set a monitor switch for all partitions, issue the command:

```
db2 update monitor switches using <switch name> on global
```

To reset monitor elements to zero, thereby allowing you to start with fresh system monitor information, use the following command (which can also be qualified to occur only on particular partitions):

```
DB2 RESET MONITOR ALL | FOR DATABASE <database alias>
```

Related reference:

- Snapshot monitor
- Snapshot monitor CLP commands
- Event monitors
- Creating an event monitor

- Formatting file or pipe event monitor output from a command line
- Event monitor sample output

Configuring health indicators for health monitors

The health monitor checks the state of your system using health indicators to determine if an alert should be issued. Preconfigured actions can be taken in response to alerts. The health monitor can also log alerts in the administration notification log and send notifications by e-mail or pager. For information on how to set up health indicators via the Health Center, see Configuring health indicators using Health Center

The following example provides an overview of the steps necessary to configure a health indicator and then monitor for health alerts on that indicator from the CLP.

The scenario is that you have decided that you want to be alerted when a particular database is in a non-normal state. This will involve monitoring the health indicator ″db.db_op_status″, the database operational state health indicator.

To determine the current settings for this health indicator on the SAMPLE database, use the following command:

```
DB2 GET ALERT CFG FOR DATABASE ON SAMPLE
```

The following is the default output resulting from this request for configuration information:

```
Alert Configuration

Indicator Name            = db.db_op_status
    Default                      = Yes
    Type                           = State-based
    Sensitivity                    = 0
    Formula                        = db.db_status;
    Actions                        = Disabled
    Threshold or State checking    = Enabled
...
```

Thus the state checking is enabled for this health indicator. If you wanted to alter any of these fields, you could do so using the command UPDATE ALERT CFG, for example:

```
DB2 UPDATE ALERT CFG FOR DATABASE ON SAMPLE using db.db_op_status SET THRESHOLDSCHECKED NO
```

To return to the default values, use the RESET ALERT CFG command, for example:

```
DB2 RESET ALERT CFG FOR DATABASE ON SAMPLE USING db.db_op_status
```

Note: The table space has to be in the non-normal state when the health indicator evaluates in order to generate the alert. So, if the state doesn’t persist long enough, then it’s possible that an alert will not be generated.

In DB2 version 8.2 FixPak 7, the refresh interval for this particular health indicator is 5 minutes. Therefore if you want to test this alert situation, perform an action that will put the database in a non-normal state (for example by quiescing it) and then wait 5 minutes. Unfortunately, there is no method for you to determine the refresh intervals. Most of them range between 5 and 60 minutes.

Once that time has passed, obtain a health monitor snapshot:

```
DB2 CONNECT TO sample
DB2 QUIESCE DATABASE IMMEDIATE
<Wait **>
DB2 GET HEALTH SNAPSHOT FOR DATABASE ON sample

            Database Health Snapshot
Snapshot timestamp        = 01/14/2005 15:02:03.96836
Database name         = SAMPLE
Database path         = D:\DB2\NODE0000\SQL00002\
Input database alias       = SAMPLE
Operating system running at database server  = NT
Location of the database      = Local
Database highest severity alert state   = Attention

Health Indicators:
     Indicator Name     = db.db_op_status
Value                         = 2
Evaluation timestamp  = 01/14/2005 15:00:27.296000
Alert state                             = Attention
```

If you were unaware that the database was quiesced, you could use the GET RECOMMENDATIONS command in order to obtain the recommended actions for this particular alert situation:

```
DB2 GET RECOMMENDATIONS FOR HEALTH INDICATOR db.db_op_status FOR DATABASE ON sample

Problem:
    Indicator Name                  = db.db_op_status
       Value                                   = 2
       Evaluation timestamp   = 01/14/2005 15:05:27
       Alert state                          = Attention
       Partition                               = 0
       Additional information                  =

Recommendations:

Recommendation: Unquiesce the database.
Rank: 1
...
```

If you have quiesced your database to perform this test, unquiesce it now using any one of the methods described in the GET RECOMMENDATIONS output. For example:

```
DB2 UNQUIESCE DATABASE
DB2 CONNECT RESET
```

Related reference:

- Introduction to the health monitor
- Health monitor interfaces
- Health indicator configuration
- Health monitor sample output

Health monitor output

The two pieces of information that you will need in order to understand health monitor output are as follows:

1. The meaning of the health indicator. See Health indicators summary for definitions of all of the indicators.
2. The meaning of the attribute values displayed for each health indicator. Some of the attributes (evaluation flag, thresholds, sensitivity) define when the health

monitor will generate an alert for the health indicator. Other attributes (action flag, actions) define what the health monitor does upon generating the alert. For more information about these attributes, see Health indicator configuration.

Two examples of health snapshot output and their meaning are provided here:

```
Health monitor sample output.
```

Interpreting event monitor and snapshot output

To begin interpreting system monitor output, you must first know how to find the meaning of the various monitor elements. For definitions of monitor elements that can be referred by snapshot monitoring, refer to Snapshot monitor logical data groups and monitor elements. For definitions of monitor elements that can be logged by event monitors, refer to Event monitor logical data groups and monitor elements.

Diagnosing the source of the problem

Diagnostic tools

7 db2diag - Analyzing db2diag.log files

7 In DB2 version 8, the primary log file intended for use by database and system
7 administrators is the Administration Notification log. The db2diag.log file, on the
7 other hand, is intended for use by DB2 customer support for troubleshooting
7 purposes.

7 The db2diag tool serves to filter and format the volume of information available in
7 the db2diag.log.

7 **Example 1**: Filtering the db2diag.log by database name:

7 If there are several databases in the instance, and you wish to only see those
7 messages which pertain to the database "SAMPLE", you can filter the db2diag.log
7 as follows:

```
7 db2diag -g db=SAMPLE
```

7 Thus you would only see db2diag.log records which contained "DB: SAMPLE",
7 such as:

```
7 2005-01-14-15.16.52.984000-480 E105657825H410     LEVEL: Error
7 PID     : 2200                 TID  : 2148        PROC : db2syscs.exe
7 INSTANCE: DB2                  NODE : 000         DB   : SAMPLE
7 APPHDL  : 0-254                APPID: *LOCAL.DB2.050114231611
7 FUNCTION: DB2 UDB, base sys utilities, sqleDatabaseUnquiesce, probe:2
7 MESSAGE : ADM7509W  Database unquiesce request has completed successfully.
```

7 **Example 2**: Filtering the db2diag.log by process id:

7 The following command can be used to display all severe error messages produced
7 by processes running on partitions 0,1,2, or 3 with the process ID (PID) 2200:

```
7         db2diag -g level=Severe,pid=2200 -n 0,1,2,3
```

7 Note that this command could have been written a couple of different ways,
7 including "db2diag -l severe -pid 2200 -n 0,1,2,3". These commands would
7 successfully retrieve db2diag.log records which meet these requirements, such as:

```
7   2005-01-14-12.18.51.515000-480 I105644317H448     LEVEL: Severe
7   PID     : 2200                 TID  : 2108       PROC : db2syscs.exe
7   INSTANCE: DB2                  NODE : 000        DB   : SAMPLE
7   APPHDL  : 0-47                 APPID: *LOCAL.DB2.050114195055
7   FUNCTION: DB2 UDB, catalog services, sqlrldre, probe:60
7   RETCODE : ZRC=0x800D002D=-2146631635=SQLM_RC_EVACT "drop active mon"
7             DIA8053C Event monitor cannot be deactivated.
```

7 **Example 3**: Formatting the db2diag tool output.

7 The following command filters all records occurring after January 1, 2005
7 containing non-severe and severe errors logged on partitions 0,1 or 2. It outputs
7 the matched records such that the time stamp, partition number and level appear
7 on the first line, pid, tid and instance name on the second line, and the error
7 message follows thereafter:

```
7   db2diag -time 2005-01-01 -node "0,1,2" -level "Severe, Error"
7   Partition: %node Message Level: %{level} \nPid: %{pid}  Tid: %{tid} Instance: %{instance}\nMessage: (
```

7 An example of the output produced is as follows:

```
7   Time: 2005-01-05-14.59.58.984000 Partition: 000 Message Level: Severe
7   Pid: 2616 Tid: 2452 Instance: DB2
7   Message: Backup Terminated.
```

7 For more information, issue the following commands:

7 - **db2diag -help** provides a short description of the options
7 - **db2diag -h brief** provides descriptions for all options without examples
7 - **db2diag -h notes** provides usage notes and restrictions
7 - **db2diag -h examples** provides a small set of examples to get started
7 - **db2diag -h tutorial** provides examples for all available options
7 - **db2diag -h all** provides the most complete list of options

7 **Related reference:**

7 - db2diag - db2diag.log analysis tool Command
7 - Administration logs, error logs and first failure data capture

db2greg - Displaying and altering the Global Registry (UNIX)

The Global Registry lives in **/var/db2/global.reg** (/var/opt/db2/global.reg on HP-UX) and only exists on UNIX and Linux platforms. It consists of three different record types:

- ″Service″: Service records contain information at the product level - i.e. version, install path, etc.
- ″Instance″: Instance records contain information at the instance level - i.e. Instance name, instance path, version, the ″start-at-boot″ flag, etc.
- ″Variable″: Variable records contain information at the variable level - i.e. Variable name, Variable value, and Comment

One can view (and manipulate with root access) the Global Registry with the ″db2greg″ tool. This tool is located in sqllib/bin, and in the install directory under bin as well (for use when logged in as root). You should only use this tool if requested to do so by DB2 Customer Support.

db2look - Mimicking databases

There are many times when it is advantageous to be able to create a database that is similar in structure to another database. For example, rather than testing out new applications or recovery plans on a production system, it makes more sense to create a test system that is similar in structure and data, and to then do the tests against it instead. This way, the production system will not be affected by the adverse performance impact of the tests or by the accidental destruction of data by an errant application. Also, when you are investigating a problem (such as invalid results, performance issues, and so on), it may be easier to debug the problem on a test system that is identical to the production system.

You can use the **db2look** tool to extract the required DDL statements needed to reproduce the database objects of one database in another database. The tool can also generate the required SQL statements needed to replicate the statistics from the one database to the other, as well as the statements needed to replicate the database configuration, database manager configuration, and registry variables. This is important because the new database may not contain the exact same set of data as the original database but you may still want the same access plans chosen for the two systems.

The db2look tool is described in detail in the *DB2 Command Reference* but you can view the list of options by executing the tool without any parameters. A more detailed usage can be displayed using the `-h` option.

Using db2look to mimic the tables in a database:

To extract the DDL for the tables in the database, use the -e option. For example, create a copy of the SAMPLE database called SAMPLE2 such that all of the objects in the first database are created in the new database:

```
C:\>db2 create database sample2
DB20000I The CREATE DATABASE command completed successfully.
C:\>db2look -d sample -e > sample.ddl
-- USER is:
-- Creating DDL for table(s)
-- Binding package automatically ...
-- Bind is successful
-- Binding package automatically ...
-- Bind is successful
```

Note: If you want the DDL for the user-defined spaces, database partition groups and buffer pools to be produced as well, add the ″`-l`″ flag after ″`-e`″ in the command above. The default node groups, buffer pools, and table spaces will not be extracted. This is because they already exist in every database by default. If you wish to mimic these, you must alter them yourself manually.

Bring up the file sample.ddl in a text editor. Because you want to execute the DDL in this file against the new database, you must change the CONNECT TO SAMPLE statement to CONNECT TO SAMPLE2. If you used the ″-l″ option, you may need to alter the path associated with the tablespace commands, such that they point to appropriate paths as well. While you are at it, take a look at the rest of the contents of the file. You should see CREATE TABLE, ALTER TABLE, and CREATE INDEX statements for all of the user tables in the sample database:

```
...
------------------------------------------------
-- DDL Statements for table "DB2"."ORG"
------------------------------------------------

 CREATE TABLE "DB2"."ORG"  (
    "DEPTNUMB" SMALLINT NOT NULL ,
    "DEPTNAME" VARCHAR(14) ,
    "MANAGER" SMALLINT ,
    "DIVISION" VARCHAR(10) ,
    "LOCATION" VARCHAR(13) )
   IN "USERSPACE1" ;
...
```

Once you have changed the connect statement, execute the statements, as follows:

```
C:\>db2 -tvf sample.ddl > sample2.out
```

Take a look at the sample2.out output file -- everything should have been executed successfully. If errors have occurred, the error messages should state what the problem is. Fix those problems and execute the statements again.

As you can see in the output, DDL for all of the user tables are exported. This is the default behavior but there are other options available to be more specific about the tables included. For example, to only include the STAFF and ORG tables, use the -t option:

```
C:\>db2look -d sample -e -t staff org > staff_org.ddl
```

To only include tables with the schema DB2, use the -z option:

```
C:\>db2look -d sample -e -z db2 > db2.ddl
```

Mimicking statistics for tables:

If the intent of the test database is to do performance testing or to debug a performance problem, it is essential that access plans generated for both databases are identical. The optimizer generates access plans based on statistics, configuration parameters, registry variables, and environment variables. If these things are identical between the two systems then it is very likely that the access plans will be the same.

If both databases have the exact same data loaded into them and runstats is performed on both, the statistics should be identical. However, if the databases contain different data or if only a subset of data is being used in the test database then the statistics will likely be very different. In such a case, you can use db2look to gather the statistics from the production database and place them into the test database. This is done by creating UPDATE statements against the SYSSTAT set of updatable catalog tables as well as RUNSTATS commands against all of the tables.

The option for creating the statistic statements is -m. Going back to the SAMPLE/SAMPLE2 example, gather the statistics from SAMPLE and add them into SAMPLE2:

```
C:\>db2look -d sample -m > stats.dml
-- USER is:
-- Running db2look in mimic mode
```

As before, the output file must be edited such that the CONNECT TO SAMPLE statement is changed to CONNECT TO SAMPLE2. Again, take a look at the rest of the file to see what some of the runstats and update statements look like:

```
...
-- Mimic table ORG
RUNSTATS ON TABLE "DB2"."ORG" ;

UPDATE SYSSTAT.INDEXES
SET NLEAF=-1,
    NLEVELS=-1,
    FIRSTKEYCARD=-1,
    FIRST2KEYCARD=-1,
    FIRST3KEYCARD=-1,
    FIRST4KEYCARD=-1,
    FULLKEYCARD=-1,
    CLUSTERFACTOR=-1,
    CLUSTERRATIO=-1,
    SEQUENTIAL_PAGES=-1,
    PAGE_FETCH_PAIRS='',
    DENSITY=-1,
    AVERAGE_SEQUENCE_GAP=-1,
    AVERAGE_SEQUENCE_FETCH_GAP=-1,
    AVERAGE_SEQUENCE_PAGES=-1,
    AVERAGE_SEQUENCE_FETCH_PAGES=-1,
    AVERAGE_RANDOM_PAGES=-1,
    AVERAGE_RANDOM_FETCH_PAGES=-1,
    NUMRIDS=-1,
    NUMRIDS_DELETED=-1,
    NUM_EMPTY_LEAFS=-1
WHERE TABNAME = 'ORG' AND TABSCHEMA = 'DB2   ';
...
```

As with the -e option that extracts the DDL, the -t and -z options can be used to specify a set of tables.

Extracting configuration parameters and environment variables:

The optimizer chooses plans based on statistics, configuration parameters, registry variables, and environment variables. As with the statistics, db2look can be used to generate the necessary configuration update and set statements. This is done using the -f option. For example:

```
C:\>db2look -d sample -f > config.txt

-- USER is:
-- This CLP file was created using DB2LOOK Version 8.2
-- Timestamp: 2/16/2005 1:26:50 AM
-- Database Name: SAMPLE
-- Database Manager Version: DB2/NT Version 8.2.1
-- Database Codepage: 1252
-- Database Collating Sequence is: UNIQUE

-- Binding package automatically ...
-- Bind is successful
-- Binding package automatically ...
-- Bind is successful

CONNECT TO SAMPLE;

--------------------------------------------------------
-- Database and Database Manager configuration parameters
--------------------------------------------------------

UPDATE DBM CFG USING cpuspeed 3.739392e-007;
UPDATE DBM CFG USING intra_parallel NO;
UPDATE DBM CFG USING comm_bandwidth 1.000000;
UPDATE DBM CFG USING federated NO;
UPDATE DBM CFG USING fed_noauth NO;

UPDATE DB CFG FOR SAMPLE USING locklist 50;
UPDATE DB CFG FOR SAMPLE USING dft_degree 1;
UPDATE DB CFG FOR SAMPLE USING maxlocks 22;
UPDATE DB CFG FOR SAMPLE USING avg_appls 1;
UPDATE DB CFG FOR SAMPLE USING stmtheap 2048;
UPDATE DB CFG FOR SAMPLE USING dft_queryopt 5;

---------------------------------
-- Environment Variables settings
---------------------------------
```

```
COMMIT WORK;

CONNECT RESET;

TERMINATE;
```

Note: Only those parameters and variables that affect access plan generation will be included.

Related reference:

- db2look - DB2 Statistics and DDL Extraction Tool Command

db2nstck and db2_call_stack - Generating EDU call stacks

The term EDU stands for ″engine dispatch unit″ and refers to a thread (Windows) or a process (UNIX) that is doing work on behalf of DB2. The **db2nstck** (Windows) and **db2_call_stack** (UNIX) commands are used to generate call stacks for the EDUs that are running under an instance. A call stack, also known as a stack traceback, shows the processing path that an EDU is currently in. The function that the EDU is currently executing is at the top of the stack, the function that called that one is below it, and so on.

These commands are mainly used when it appears that the DB2 engine has become ″hung″. (In most cases, problems that appear to be with DB2 are in reality caused by application problems. For example, a long-running transaction may be holding locks that all other applications are waiting on. All of the applications will appear to be hung but they are really just waiting for one or more locks to be released. Situations like this can usually be identified using DB2's snapshot and event monitor tools.

Operating system commands (such as `iostat` and `ps` in UNIX) can be used to determine if EDUs are actually doing any work. If it appears that none of the EDUs are doing any processing, an engine hang may be a possibility. By generating multiple call stacks for each EDU (with a sufficient amount of time in between -- 1 to 2 minutes for example), you can compare the call stacks to see if the EDUs are in a different processing path from one stack to the next. If all of the EDUs are stuck in the same function that was shown in the previous call stack then there may in fact be a problem with the DB2 engine. In that case, the call stacks must be provided to IBM so that the hang can be analyzed.

In UNIX, the db2_call_stack command generates call stacks for both single and multi-node instances. The call stacks are placed in files in the diagnostic directory (sqllib/db2dump by default). Prior v8.2.2 each EDU has its own file which is called t*processID.nodenumber*. For v8.2.2 Each EUD has its own file which is called t<processID>.<nodenumber>.xml The files are textbased and can be viewed using any editor.

Here is an example of a portion of a call stack on UNIX:

```
*** Start stack traceback ***

0xD024E2C8 select + 0xAC
0x205E0244 sqlorqueInternal__FP13sqlo_que_descP12SQLO_MSG_HDRil + 0x100
0x205E008C sqlorque2 + 0xAC
0x10004840 sqleRunSysCtlr__Fv + 0x10C
0x10002D44 sqleSysCtlr__Fv + 0x8C
0x205C6FFC sqloRunInstance + 0x75C
```

```
0x100028EC DB2main + 0x8A4
0x10003470 main + 0xC

*** End stack traceback ***
```

Note: Running db2_call_stack will temporarily affect the performance of the server. It is often preferable to use a ″kill -36″ (signal 36 is for AIX, 21 for sun, 29 for hp, 23 for linux.) command directly against the db2sysc process on UNIX platforms. Running `ps` will show every db2 process as db2sysc. `ps -elf` will show only one db2sysc per logical partition, which is the parent of all db2 processes, and this is the process against which kill -36 should be run. See **Diagnostic tools for UNIX-based systems**.

In Windows, the db2nstck command is used for single-node instances and db2_call_stack is used for multi- node instances. All of the call stacks will be placed into a single file in the diagnostic directory (sqllib**instance** by default). The db2nstck command tells you the name of the file name. For example:

```
C:\>db2nstck
The stack dump has been saved in the file C:\SQLLIB\DB2\P2292.000
```

This file is in a binary format. To convert it to readable text, you need the db2xprt tool and the DB2 for Windows .pdb files (i.e. symbol files). See **Formatting and interpreting trap files** for information about where to find this tool and how to use it.

Related reference:

- Trap files

9 db2pd - Monitoring and troubleshooting DB2

9 The db2pd tool is a problem determination tool which contains quick and
7 immediate information from the DB2 memory sets. It is similar to the Informix
7 ″onstat″ tool, and runs very quickly as it is not acquiring any locks or latches and
7 runs outside of the DB2 engine resources.

7 The tool collects this information without acquiring any latches or using any
7 engine resources. It is therefore possible (and expected) to retrieve information that
7 is changing while db2pd is collecting information; hence the data may not be
7 completely accurate. If null pointers are encountered, a signal handler is used to
7 prevent db2pd from aborting abnormally. This can result in messages such as
7 ″`Changing data structure forced command termination`″ to appear in the output.
7 Nonetheless, the tool can be helpful for problem determination. Two benefits to
7 collecting information without latching include faster retrieval and no competition
7 for engine resources.

7 What follow are some examples of problem scenarios in which db2pd can be used
7 to expedite problem determination.

7 **Scenario 1**: Catching a lock timeout using the ″`-catch`″ db2pd option:

7 This option is UNDOCUMENTED, yet enabled on customer systems.

7 The purpose of `db2pd -catch` is to allow the user to catch any sqlcode (and reason
7 code), zrc code, or ecf code and capture the information needed to solve the error
7 code.

The primary reaction is to execute the db2cos (callout script) that can dynamically be altered to run any db2pd command, os command, or any other command needed to solve the problem. The template db2cos file is located in sqllib/cfg and must be moved to sqllib in order to be called.

Usage:

```

-catch clear | status | <errorCode> [<action>] [count=<count>]
      Sets catchFlag to catch error or warning.


        Error Codes:
        <sqlCode>[,<reasonCode>] / sqlcode=<sqlCode>[,<reasonCode>]
        ZRC (hex or integer)
        ZRC #define (such as SQLP_LTIMEOUT)
        ECF (hex or integer)
        "deadlock" or "locktimeout"

        Actions:
        [db2cos] (default)              Run sqllib/db2cos callout script
        [stopdb2trc]                    Stop db2trc
        [dumpcomponent]                 Dump component flag
        [component=<componentID>]       Component ID
        [lockname=<lockname>]           Lockname for catching specific lock
                                        (lockname=000200030000001F0000000052)
        [locktype=<locktype>]           Locktype for catching specific lock
                                        (locktype=R or locktype=52)
```

Examples (see last example for a step-by-step locktimeout catch):

1. Catch sqlcode -911 reason code 68 and run db2cos

   ```
   db2pd -catch -911,68 db2cos    OR
   db2pd -catch -911,68

   Error Catch #1
           Sqlcode:        -911
           ReasonCode:     68
           ZRC:            0
           ECF:            0
           Component ID:   0
           LockName:       Not Set
           LockType:       Not Set
           Current Count:  0
           Max Count:      255
           Bitmap:         0x261
           Action:         Error code catch flag enabled
           Action:         Execute sqllib/db2cos callout script
   ```

2. Catch sqlcode -911 (and any reason code) and run db2cos

   ```
   db2pd -catch -911

   Input ECF string '-911' parsed as 0xFFFFFC71 (-911).
   Error Catch #2
           Sqlcode:        -911
           ReasonCode:     0
           ZRC:            0
           ECF:            0
           Component ID:   0
           LockName:       Not Set
           LockType:       Not Set
           Current Count:  0
           Max Count:      255
           Bitmap:         0x61
           Action:         Error code catch flag enabled
           Action:         Execute sqllib/db2cos callout script
   ```

3. Catch SQLP_LTIMEOUT and run db2cos

   ```
   db2pd -catch SQLP_LTIMEOUT

   Input ZRC string 'SQLP_LTIMEOUT' parsed as 0x80100044 (-2146435004).
   Error Catch #3
           Sqlcode:        0
           ReasonCode:     0
           ZRC:            -2146435004
           ECF:            0
           Component ID:   0
           LockName:       Not Set
           LockType:       Not Set
           Current Count:  0
           Max Count:      255
           Bitmap:         0xA1
           Action:         Error code catch flag enabled
           Action:         Execute sqllib/db2cos callout script
   ```

4. Catch SQLP_LTIMEOUT and run db2cos, but only fire the script one time.

   ```
   db2pd -catch SQLP_LTIMEOUT count=1

   Input ZRC string 'SQLP_LTIMEOUT' parsed as 0x80100044 (-2146435004).
   Error Catch #4
           Sqlcode:        0
           ReasonCode:     0
           ZRC:            -2146435004
           ECF:            0
           Component ID:   0
           LockName:       Not Set
           LockType:       Not Set
           Current Count:  0
           Max Count:      1
           Bitmap:         0xA1
           Action:         Error code catch flag enabled
           Action:         Execute sqllib/db2cos callout script
   ```

5. Only clear -911,68 setting

   ```
   db2pd -catch clear -911,68
   Error catch setting cleared for sqlCode -911 reasonCode 68
   ```

6. Clear all settings

   ```
   db2pd -catch clear all
   All error catch flag settings cleared.
   ```

7. Step-by-step instructions to catch a lock timeout.
   - Copy the db2cos template into sqllib.

     ```
     cp sqllib/cfg/db2cos sqllib/db2cos
     ```

   - Set the error catch setting. You can use -911,68, -911, SQLP_LTIMEOUT, or locktimeout. If fact, if you know the lock type or lock name in advance, you can use use the locktype or lockname suboptions to filter out unwanted lock catches.

     ```
     db2pd -catch locktimeout count=1
     Error Catch #1
             Sqlcode:        0
             ReasonCode:     0
             ZRC:            -2146435004
             ECF:            0
             Component ID:   0
             LockName:       Not Set
             LockType:       Not Set
             Current Count:  0
             Max Count:      1
             Bitmap:         0xA1
             Action:         Error code catch flag enabled
             Action:         Execute sqllib/db2cos callout script
     ```

- Reproduce a lock timeout.

```
db2 update staff set id = 0
DB21034E  The command was processed as an SQL statement because it was not a
valid Command Line Processor command.  During SQL processing it returned:
SQL0911N  The current transaction has been rolled back because of a deadlock
or timeout.  Reason code "68".  SQLSTATE=40001
```

- Notice the db2diag.log tells us what happened.

```
2005-01-06-15.31.59.017134-300 I416442C274        LEVEL: Event
PID     : 9875676              TID  : 1           PROC : db2pd
INSTANCE: jmcmahon             NODE : 000
FUNCTION: DB2 UDB, RAS/PD component, pdErrorCatch, probe:30
START   : Error catch set for ZRC -2146435004

2005-01-06-15.36.07.084657-300 I416717C403        LEVEL: Event
PID     : 10379460             TID  : 1           PROC : db2agent (SAMPLE)
INSTANCE: jmcmahon             NODE : 000         DB   : SAMPLE
APPHDL  : 0-71                 APPID: *LOCAL.jmcmahon.050106202920
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:10
START   : Invoking sqllib/db2cos script from lock manager sqlplnfd

2005-01-06-15.36.09.274708-300 I417121C386        LEVEL: Event
PID     : 10379460             TID  : 1           PROC : db2agent (SAMPLE)
INSTANCE: jmcmahon             NODE : 000         DB   : SAMPLE
APPHDL  : 0-71                 APPID: *LOCAL.jmcmahon.050106202920
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:20
STOP    : Completed invoking sqllib/db2cos script
```

- Now look in `sqllib/db2dump/db2cos.rpt` which is the default output file that contains the db2pd output generated when the error was caught.

```
Lock Timeout Caught
Thu Jan  6 15:36:07 EST 2005
Instance         jmcmahon
Datbase:         SAMPLE
Partition Number: 0
PID:             10379460
TID:             1
Function:        sqlplnfd
Component:       lock manager
Probe:           999
Timestamp:       2005-01-06-15.36.07.084474
AppID:           *LOCAL.jmcmahon.050106202920
AppHdl:

<snip>
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:06:53

Locks:
Address    TranHdl Lockname                   Type    Mode Sts Owner Dur HldCnt   Att Rlse
0x402C6B30 3       00020003000000040000000052 Row     ..X  W*  3     1   0        0   0x40
```

- Just look for the ″W*″ as this is the lock that experienced the timeout. You can map this to a transaction, application, agent, and even SQL statement with all of the db2pd output provided. You can get strategic and narrow down the output or use other commands to collect the information you need. For example, you could change the db2pd options to use the ″`-locks wait`″ option that only prints locks with a wait status and their waiter. You could also put in `-fmtlock`, `-app`, and `-agent` options if that's what you need.

```
   "LOCKTIMEOUT")
      echo "Lock Timeout Caught"          >> $HOME/sqllib/db2dump/db2cos.rpt
      date                                >> $HOME/sqllib/db2dump/db2cos.rpt
      echo "Instance         " $instance  >> $HOME/sqllib/db2dump/db2cos.rpt
      echo "Datbase:         " $database  >> $HOME/sqllib/db2dump/db2cos.rpt
      echo "Partition Number:" $dbpart    >> $HOME/sqllib/db2dump/db2cos.rpt
      echo "PID:             " $pid       >> $HOME/sqllib/db2dump/db2cos.rpt
      echo "TID:             " $tid       >> $HOME/sqllib/db2dump/db2cos.rpt
```

```
          echo "Function:        " $function  >> $HOME/sqllib/db2dump/db2cos.rpt
          echo "Component:       " $component >> $HOME/sqllib/db2dump/db2cos.rpt
          echo "Probe:           " $probe     >> $HOME/sqllib/db2dump/db2cos.rpt
          echo "Timestamp:       " $timestamp >> $HOME/sqllib/db2dump/db2cos.rpt
          echo "AppID:           " $appid     >> $HOME/sqllib/db2dump/db2cos.rpt
          echo "AppHdl:          " $apphld    >> $HOME/sqllib/db2dump/db2cos.rpt
          db2pd -db $database                 >> $HOME/sqllib/db2dump/db2cos.rpt
          db2pd -db $database -locks wait -fmtlock -app -age >> $HOME/sqllib/db2dump/db2cos.rpt
      ;;
```

It's as simple as that!

Related reference:

- db2pd - Monitor and Troubleshoot DB2 Command

Scenario 2: Identifying a lock wait.

There are several types of locks reported with `db2pd -locks` that can cause an application to wait. Common ones are table, pool, and row where other possibilities are Internal V (dynamic sql cache), Internal P (package cache), and CatCache (catalog cache). Prior to db2pd, there was no way to identify the Internal V, Internal P, or CatCache lock.

`db2pd -locks` output showing Catcache, Row, Internal V, and Internal P.

```
Locks:
Address            TranHdl    Lockname                   Type       Mode Sts Owner
0x00000002202EA3E0 2          0000000200000B0520A4774043 CatCache   ..X  G   2
0x00000002202E30D8 2          0000000200004070000000052 Row        .NS  G   2
0x00000002202DCC28 2          0000000200000010001860056 Internal V ..S  G   2
0x00000002202D88F0 2          53514C4532453047668358B641 Internal P ..S  G   2


Dur HldCnt     Att Rlse
255 0          0   0x40
1   0          0   0x40
1   0          0   0x40
1   1          0   0x40
```

`db2pd -catalogcache` output showing the lock 0000000200000B0520A4774043 maps to a SYSTABLES entry for the SYSIBM table.

```
Catalog Cache:
Configured Size        655360
Current Size           353416
Maximum Size           4294950912
High Water Mark        393216

SYSTABLES:
Address            Schema   Name               Type TableID TbspaceID
0x0000000220A47740 SYSIBM   SYSFUNCTIONS       V    0       0

LastRefID  CatalogCacheLoadingLock    CatalogCacheUsageLock      Sts
10615      00010000000000220A4774043 0000000200000B0520A4774043 V
```

`db2pd -dynamic` output showing that lock 0000000200000010001860056 maps to an sql statement creating a view. Since the dynamic cache has three tiers, one must identify the lock in the Variation section then map the Variation to an Environment and finally map the Environment to a Statement.

```
Dynamic Cache:
Current Memory Used           1019927
Total Heap Size               1271398
Cache Overflow Flag           0
```

```
Number of References         150
Number of Statement Inserts   77
Number of Statement Deletes   1
Number of Variation Inserts   150
Number of Statements         76

Dynamic SQL Statements:
Address            AnchID StmtUID    NumEnv     NumVar     NumRef     NumExe     Text
0x0000000220ACF220 134    2          2          2          2          1          create view
syscat.librarybindfiles (libschema, libname, libversion, package_schema,
package_name, package_version, bindfile_path)  as select
  l.libschema, l.libname, b.libversion, b.package_schema, b.package_name, b.package_version,
  b.bindfile_path  from sysibm.syslibrarybindfiles as b,
  sysibm.syslibraries as l  where b.lib_id = l.lib_id

Dynamic SQL Environments:
Address            AnchID StmtUID    EnvID Iso QOpt Blk
0x0000000220ACF480 134    2          1     CS  5    U
0x0000000220ACFE60 134    2          2     CS  5    U

Dynamic SQL Variations:
Address            AnchID StmtUID    EnvID VarID      NumRef     Typ Lockname
0x0000000220ACF6C0 134    2          1     1          1          1   00000002000000010001860056
```

db2pd -static output showing that lock 53514C4532453047668358B641 maps to a Package SQLE2E0G.

```
Static Cache:
Current Memory Used          991807
Total Heap Size              1271398
Cache Overflow Flag          0
Number of References         3
Number of Package Inserts    2
Number of Section Inserts    0

Packages:
Address            Schema   PkgName  Version             UniqueID NumSec
0x0000000220A01780 JMCMAHON SQLE2E0G                     AAAAARAU 0

UseCount   NumRef     Iso QOpt  Blk Lockname
0          1          CS  5     U   53514C4532453047668358B641
```

Scenario 3: Mapping an application to a dynamic SQL statement.

Starting with Saturn, db2pd -applications reports the Current and Last Anchor ID and Statement Unique ID for dynamic SQL statements. This allows direct mapping from an application to a dynamic sql statements.

```
db2pd -app -dyn

Applications:
Address            AppHandl [nod-index] NumAgents  CoorPid    Status
0x00000002006D2120 780      [000-00780] 1          10615      UOW-Executing

C-AnchID C-StmtUID  L-AnchID L-StmtUID  Appid
163      1          110      1          *LOCAL.jmcmahon.050202200412

Dynamic SQL Statements:
Address            AnchID StmtUID    NumEnv     NumVar   NumRef   NumExe  Text
0x0000000220A02760 163    1          2          2        2        1       CREATE VIEW MYVIEW
0x0000000220A0B460 110    1          2          2        2        1       CREATE VIEW YOURVIEW
```

Scenario 4: Monitoring memory usage.

7 The db2pd -memsets and -mempools options report statistics about DB2 Memory
7 Sets and Memory Pools which can be very useful when trying to understand
7 memory usage.

```
7 Memory Sets:
7 Name         Address            Id          Size             Key      DBP    Type   Ov OvSize
7 SAMPLE       0x0000000220000000 290831      46055424         0x0      0      1      Y  8978432
7 App780       0xFFFFFFFF77F00000 1487884     131072           0x0      0      3      N  0
7
7 Memory Pools:
7 Address            MemSet   PoolName   Id    Overhead   LogSz   LogUpBnd     LogHWM
7 0x0000000220000CB0 SAMPLE   utilh      5     0          304     20496384     304
7 0x0000000220000BB0 SAMPLE   undefh     59    0          0       12345678     0
7 0x0000000220000AB0 SAMPLE   pckcacheh  7     30592      768781  Unlimited    768781
7 0x00000002200009B0 SAMPLE   catcacheh  8     0          194696  Unlimited    194696
7 0x00000002200008B0 SAMPLE   bph        16    12064      4265856 Unlimited    4265856
7 0x00000002200007B0 SAMPLE   bph        16    32         528832  Unlimited    528832
7 0x00000002200006B0 SAMPLE   bph        16    32         266688  Unlimited    266688
7 0x00000002200005B0 SAMPLE   bph        16    32         135616  Unlimited    135616
7 0x00000002200004B0 SAMPLE   bph        16    32         70080   Unlimited    70080
7 0x00000002200003B0 SAMPLE   lockh      4     0          492160  524288       492160
7 0x00000002200002B0 SAMPLE   dbh        2     59168      3800760 8683520      3800760
7
7 PhySz               PhyUpBnd             PhyHWM               Bnd BlkCnt CfgParm
7 16384               20496384             16384                Ovf  0     UTIL_HEAP_SZ
7 0                   12353536             0                    Ovf  0     n/a
7 819200              Unlimited            819200               Ovf  0     PCKCACHESZ
7 262144              Unlimited            262144               Ovf  0     CATALOGCACHE_SZ
7 4292608             Unlimited            4292608              Ovf  0     n/a
7 540672              Unlimited            540672               Ovf  0     n/a
7 278528              Unlimited            278528               Ovf  0     n/a
7 147456              Unlimited            147456               Ovf  0     n/a
7 81920               Unlimited            81920                Ovf  0     n/a
7 524288              524288               524288               Ovf  0     LOCKLIST
7 3932160             8683520              3932160              Ovf  0     DBHEAP
```

7 **Scnenario 5**: Memory block usage using FFDC feature. (This is undocumented, yet
7 enabled on customer systems). You must set DB2MEMDBG=FFDC to enable this feature.

7 **Usage**

```
7    -memblock [dbms, fcm, fmp, appctl <id>, private (windows)] [scrape (unix)]
```

7 This option will report extended memory blocks from the following
7 sets/pools/heaps:

7 • DBMS
7 • FCM
7 • FMP
7 • AppCtl (shared memory set id must be supplied and database must be
7 supplied)
7 • Database
7 • Private (windows only)
7 • The ″scrape″ option on Unix does not use OSS calls to search for blocks. Instead,
7 it starts at the beginning of the memory set and walks it to the end, 4 bytes at a
7 time. This can be useful if there is suspicion that a memory block has been
7 ″orphaned″ by the OSS layer.

7 First, turn on FFDC extended memory blocks:

```
7        db2set DB2MEMDBG=FFDC
```

7 Example for DBMS set:

```
db2pd -memb OR
db2pd -memb dbms
Printing all blocks in DBMS set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC    File
0x30944048 62         resynch      2          92         1 0 0 2314  1219138542
0x3092AF98 62         resynch      1          102444     1 0 0 124   4075170607
0x30109748 57         ostrack      6          4860032    1 0 0 2987  4279232714
0x300E5CC8 57         ostrack      5          140032     1 0 0 2974  4279232714
0x300E0048 57         ostrack      1          64         1 0 0 2910  4279232714
0x300E00A8 57         ostrack      2          208        1 0 0 2923  4279232714
0x300E0198 57         ostrack      3          80         1 0 0 2939  4279232714
0x300E0208 57         ostrack      4          80         1 0 0 2949  4279232714
0x30924048 70         apmh         5          16016      1 0 0 141   2389885231
0x30927EF8 70         apmh         6          124        1 0 0 2760  1219138542
```

You can then use the `db2fntohash` tool to map the File hash value to a filename.

If engn/pd/db2filenames.lst does NOT exist, you can create it. This should be built with each nightly build. If engn/pd/db2filenames.lst DOES exist, `db2fntohash -l <hashvalue>` will provide the filename.

```
db2fntohash db2filenames.in db2filenames.lst

db2fntohash -l 1219138542 db2filenames.lst
db2fntohash: hash value "1219138542" => file name "sqlesysc.C"

db2fntohash -l 4075170607 db2filenames.lst
db2fntohash: hash value "4075170607" => file name "sqlerlst.C"

db2fntohash -l 4279232714 db2filenames.lst
db2fntohash: hash value "4279232714" => file name "startdbm.C"
```

Example for FCM set:

```
db2pd -memb fcm
Printing all blocks in FCM set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC    File
0x813C4048 79         fcmrqb       19         16400      1 0 0 2547  1272798017
0x813B8048 79         fcmrqb       18         47120      1 0 0 2375  1272798017
0x813AC048 79         fcmrqb       17         47120      1 0 0 2375  1272798017
0x813A0048 79         fcmrqb       16         47120      1 0 0 2375  1272798017
```

Example for FMP set:

```
db2pd -memb fmp
Printing all blocks in FMP set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC    File
0x90009FA8 59         undefh       1          122916     1 0 0 357   4133190529
```

Example for DB set:

```
db2pd -memb -db sample
Printing all blocks in Database SAMPLE set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC    File
0x4093C048 5          utilh        1          64         1 0 0 150   3032704123
0x4093C0A8 5          utilh        2          232        1 0 0 194   1779766316
0x40940048 7          pckcacheh    2          408        1 0 0 1082  19703507
0x409401F8 7          pckcacheh    3          1060       1 0 0 2119  3048315900
0x40940738 7          pckcacheh    14         408        1 0 0 1082  19703507
```

Example for AppCtl:

```
db2pd -memb appctl -db sample
Printing all blocks in Appctl set id 566624315.
Address    PoolID     PoolName     BlockID    Size       I P N LOC    File
0xA370C048 3          appctlh      1          64         1 0 0 150   3032704123
0xA370C0A8 3          appctlh      2          2992       1 0 0 1413  2658538710
```

```
0xA370CC78 3          appctlh      3          88         1 0 0 2547  1436638871
0xA370D1E8 3          appctlh      8          144        1 0 0 778   91986655
0xA370D298 3          appctlh      9          400        1 0 0 4284  2614547685
```

Example for all instance scope sets (DBMS, FCM, FMP):

```
db2pd -memb all
Printing all blocks in DBMS set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0x30980048 74         fcm          4          16400      1 0 0 1224  2927312349
0x3091FFB8 74         fcm          2          393232     1 0 0 1009  2927312349
0x30914048 74         fcm          1          4112       1 0 0 818   1272798017
0x30915078 74         fcm          3          4112       1 0 0 1042  2927312349
0x309D6F88 11         monh         80         69699      1 0 0 772   3881733182
0x309C2F88 11         monh         79         69699      1 0 0 772   3881733182
<snip>


Printing all blocks in FCM set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0x813C4048 79         fcmrqb       19         16400      1 0 0 2547  1272798017
0x813B8048 79         fcmrqb       18         47120      1 0 0 2375  1272798017
0x813AC048 79         fcmrqb       17         47120      1 0 0 2375  1272798017
<snip>


Printing all blocks in FMP set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0x90009FA8 59         undefh       1          122916     1 0 0 357   4133190529
```

Example for all instance and database scope sets (DBMS, FCM, FMP, DB, AppCtl):

```
db2pd -memb all -db sample
Printing all blocks in DBMS set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0x30980048 74         fcm          4          16400      1 0 0 1224  2927312349
0x3091FFB8 74         fcm          2          393232     1 0 0 1009  2927312349
0x30914048 74         fcm          1          4112       1 0 0 818   1272798017
0x30915078 74         fcm          3          4112       1 0 0 1042  2927312349
0x309D6F88 11         monh         80         69699      1 0 0 772   3881733182
0x309C2F88 11         monh         79         69699      1 0 0 772   3881733182
0x309BC048 11         monh         70         184        1 0 0 130   88219158
<snip>


Printing all blocks in FCM set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0x813C4048 79         fcmrqb       19         16400      1 0 0 2547  1272798017
0x813B8048 79         fcmrqb       18         47120      1 0 0 2375  1272798017
0x813AC048 79         fcmrqb       17         47120      1 0 0 2375  1272798017
<snip>


Printing all blocks in FMP set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0x90009FA8 59         undefh       1          122916     1 0 0 357   4133190529


Printing all blocks in Appctl set id 566624315.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0xA370C048 3          appctlh      1          64         1 0 0 150   3032704123
0xA370C0A8 3          appctlh      2          2992       1 0 0 1413  2658538710
<snip>


Printing all blocks in Database SAMPLE set.
Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
0x4093C048 5          utilh        1          64         1 0 0 150   3032704123
```

```
7   0x4093C0A8 5         utilh       2          232        1 0 0 194   1779766316
7   0x40940048 7         pckcacheh   2          408        1 0 0 1082  19703507
7   0x409401F8 7         pckcacheh   3          1060       1 0 0 2119  3048315900
```

7 Example for AppCtl sets (database required):

```
7   db2pd -db sample -memb appctl
7   Printing all blocks in Appctl set id 566624315.
7   Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
7   0xA370C048 3          appctlh      1          64         1 0 0 150   3032704123
7   0xA370C0A8 3          appctlh      2          2992       1 0 0 1413  2658538710
7   0xA370CC78 3          appctlh      3          88         1 0 0 2547  1436638871
7   0xA370D1E8 3          appctlh      8          144        1 0 0 778   91986655
7   0xA370D298 3          appctlh      9          400        1 0 0 4284  2614547685
7   <snip>
7
7
7   Printing all blocks in Database SAMPLE set.
7   Address    PoolID     PoolName     BlockID    Size       I P N LOC   File
7   0x4093C048 5          utilh        1          64         1 0 0 150   3032704123
7   0x4093C0A8 5          utilh        2          232        1 0 0 194   1779766316
7   0x40940048 7          pckcacheh    2          408        1 0 0 1082  19703507
7   0x409401F8 7          pckcacheh    3          1060       1 0 0 2119  3048315900
7   0x40940738 7          pckcacheh    14         408        1 0 0 1082  19703507
```

7 **Scenario 6**: Producing a shared memory dump with db2pd -bindump
7 (undocumented, yet enabled on customer systems). Reading the shared memory
7 dump (and now core files with Saturn) using db2pd -binload (internal option).

7 **Usage**

```
7      -bindump [-db <database>] [-inst]
```

- 7 DBMS, FCM, FMP sets dumped with -bindump.
- 7 DBMS, DB, AppCtl sets dumped with -db <database>.
- 7 DBMS, FCM, FMP, DB, AppCtl sets dumped with -db <database> and -inst.

7 Example -bindump:

```
7   db2pd -bind
7   Sending -bindump output to /home/jmcmahon/db2pd.bin
7   Dumping DBMS set starting at 0x30000000 with size 52690944
7   Dumping FCM set starting at 0x80000000 with size 43270144
7   Dumping FMP set starting at 0x90000000 with size 257654784
```

7 Example -bindump -db sample:

```
7   db2pd -bind -db sample
7   Sending -bindump output to /home/jmcmahon/db2pd.bin
7   Dumping DBMS set starting at 0x30000000 with size 52690944
7   Dumping DB set for database SAMPLE starting at 0x40000000 with size 65716224
7   Dumping AppCtl set starting at 0xa0000000 with size 192839680
```

7 Example -bindump -db sample -inst:

```
7   db2pd -bind -db sample -inst
7   Sending -bindump output to /home/jmcmahon/db2pd.bin
7   Dumping DBMS set starting at 0x30000000 with size 52690944
7   Dumping DB set for database SAMPLE starting at 0x40000000 with size 65716224
7   Dumping AppCtl set starting at 0xa0000000 with size 192839680
7   Dumping FCM set starting at 0x80000000 with size 43270144
7   Dumping FMP set starting at 0x90000000 with size 257654784
```

7 **Usage**

```
7      -binload <shared memory dump> | <corefile> [any db2pd option]
```

This is very useful when you have a shared memory dump from the -bindump option. You can run ANY db2pd option just like it is from a live system.

Note that the default name is ″db2pd.bin″ which is the default when the -bindump option is used without a filename.

Starting with Saturn code (FP9), the -binload option will look for the string ″core″ and treat the file as a core dump if the filename starts with ″core.″ This functionality has only had limited testing. Please report any issues to db2raspdcore@ca.ibm.com.

Note: Run db2pd in interactive mode and then run each desired option. That will prevent having to load the shared memory dump each time you run db2pd with the -binload option.

Example -binload:

```
db2pd -binl -memp
Reading memory dump from file db2pd.bin.
Allocated segment at address 0x30000000 with shmid 524307.
Allocated segment at address 0x40000000 with shmid 393245.
Allocated segment at address 0xa0000000 with shmid 524306.
Allocated segment at address 0x80000000 with shmid 917519.
Allocated segment at address 0x90000000 with shmid 1572876.


Database Partition 0 -- Active -- Up 0 days 02:55:14


Memory Pools:
Address    MemSet   PoolName   Id    Overhead   LogSz      LogUpBnd
0x30000840 DBMS     fcm        74    16240      417792     450560
0x300007A0 DBMS     monh       11    24192      142032     368640
0x30000700 DBMS     resynch    62    12112      102504     3244032
0x30000660 DBMS     ostrack    57    13184      5000400    5046272
0x300005C0 DBMS     apmh       70    0          38730      2129920
0x30000520 DBMS     kerh       52    32         31488      376832
0x30000480 DBMS     bsuh       71    3360       3598864    17137664
0x300003E0 DBMS     sqlch      50    0          762400     786432
0x30000340 DBMS     pmth       80    0          72         216540
0x300002A0 DBMS     krcbh      69    0          19228      32768
0x30000200 DBMS     debug      87    16240      12353536   12353552
0x80000700 FCM      fcmrqb     79    16240      835632     1548336
0x80000660 FCM      fcmce      78    16240      225328     696368
0x800005C0 FCM      fcma       75    16240      237616     712752
0x80000520 FCM      fcmrqb     79    16240      835632     1548336
0x80000480 FCM      fcmce      78    16240      225328     696368
0x800003E0 FCM      fcma       75    16240      237616     712752
0x80000340 FCM      fcmbp      13    404848     17011260   17980332
0x800002A0 FCM      debug      87    16240      12353536   12517392
0x80000200 FCM      eduah      72    5984       256024     419888
0x900002A0 FMP      undefh     59    8032       122900     245163840
0x90000200 FMP      debug      87    16240      12353536   12517392

LogHWM     PhySz      PhyUpBnd   PhyHWM     Bnd BlkCnt CfgParm
417792     458752     458752     458752     Ovf 4      n/a
142116     180224     376832     180224     Ovf 13     MON_HEAP_SZ
102504     131072     3244032    131072     Ovf 2      n/a
5000400    5029888    5046272    5029888    Ovf 6      n/a
39010      49152      2129920    49152      Ovf 21     n/a
32396      49152      376832     49152      Ovf 11     n/a
3605820    3620864    17137664   3620864    Ovf 55     n/a
762400     786432     786432     786432     Ovf 163    n/a
252        16384      229376     16384      Ovf 2      n/a
19228      32768      32768      32768      Ovf 3      n/a
```

```
12353536   12369920   12369920   12369920   Ovf 1      n/a
835632     917504     1556480    917504     Phy 19     n/a
225328     294912     704512     294912     Phy 15     n/a
237616     294912     720896     294912     Phy 15     n/a
835632     917504     1556480    917504     Phy 19     n/a
225328     294912     704512     294912     Phy 15     n/a
237616     294912     720896     294912     Phy 15     n/a
17011260   17448960   17989632   17448960   Phy 38     n/a
12353536   12369920   12533760   12369920   Phy 1      n/a
256024     262144     425984     262144     Phy 1      n/a
122900     131072     245170176  131072     Phy 1      n/a
12353536   12369920   12533760   12369920   Phy 1      n/a

Removed segment with shmid 524307.
Removed segment with shmid 393245.
Removed segment with shmid 524306.
Removed segment with shmid 917519.
Removed segment with shmid 1572876.
```

Example -binload -db sample:

```
db2pd -binl -memp -db sample
Reading memory dump from file db2pd.bin.
Allocated segment at address 0x30000000 with shmid 655378.
Allocated segment at address 0x40000000 with shmid 1703948.
Allocated segment at address 0xa0000000 with shmid 1048591.
Allocated segment at address 0x80000000 with shmid 524317.
Allocated segment at address 0x90000000 with shmid 655379.


Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 02:14:42


Memory Pools:
Address    MemSet   PoolName   Id    Overhead   LogSz      LogUpBnd
0x400008E0 SAMPLE   utilh      5     0          264        20496384
0x40000840 SAMPLE   undefh     59    0          0          12345678
0x400007A0 SAMPLE   pckcacheh  7     17696      205415     Unlimited
0x40000700 SAMPLE   catcacheh  8     0          43232      Unlimited
0x40000660 SAMPLE   bph        16    11344      4238784    Unlimited
0x400005C0 SAMPLE   bph        16    112        533248     Unlimited
0x40000520 SAMPLE   bph        16    112        271104     Unlimited
0x40000480 SAMPLE   bph        16    112        140032     Unlimited
0x400003E0 SAMPLE   bph        16    112        74496      Unlimited
0x40000340 SAMPLE   lockh      4     32         475568     507904
0x400002A0 SAMPLE   dbh        2     32816      3603997    10715136
0x40000200 SAMPLE   debug      87    16240      12353536   12353552
0xA0000480 AppCtl   appctlh    3     0          48         793018
0xA00003E0 AppCtl   appctlh    3     112        14019      793018
0xA0000340 AppCtl   agsh       17    400        38276      114851888
0xA00002A0 AppCtl   mbosh      84    0          24         163864
0xA0000200 AppCtl   debug      87    16240      12353536   12517392


LogHWM     PhySz      PhyUpBnd   PhyHWM     Bnd   BlkCnt CfgParm
1560       16384      20496384   16384      Ovf   2      UTIL_HEAP_SZ
0          0          12353536   0          Ovf   0      n/a
205415     278528     Unlimited  278528     Ovf   16     PCKCACHESZ
43232      65536      Unlimited  65536      Ovf   8      CATALOGCACHE_SZ
4238784    4276224    Unlimited  4276224    Ovf   2      n/a
533248     557056     Unlimited  557056     Ovf   2      n/a
271104     294912     Unlimited  294912     Ovf   2      n/a
140032     163840     Unlimited  163840     Ovf   2      n/a
74496      98304      Unlimited  98304      Ovf   2      n/a
475568     507904     507904     507904     Ovf   2      LOCKLIST
4696564    3670016    10715136   4849664    Ovf   326    DBHEAP
12353536   12369920   12369920   12369920   Ovf   1      n/a
```

```
10116      16384      802816     16384      Phy   1      APP_CTL_HEAP_SZ
18449      327680     802816     327680     Phy   34     APP_CTL_HEAP_SZ
38276      57344000   114868224  57344000   Log   26     APPGROUP_SHARE_HEAP
24         16384      180224     16384      Phy   1      n/a
12353536   12369920   12533760   12369920   Phy   1      n/a


Removed segment with shmid 655378.
Removed segment with shmid 1703948.
Removed segment with shmid 1048591.
Removed segment with shmid 524317.
Removed segment with shmid 655379.
```

Example -binload interactive mode:

```
db2pd
db2pd> You are running db2pd in interactive mode.
db2pd> If you want command line mode, rerun db2pd with valid options.
db2pd> Type -h or -help for help.
db2pd> Type q to quit.
db2pd> -binl
Reading memory dump from file db2pd.bin.
Allocated segment at address 0x30000000 with shmid 786451.
Allocated segment at address 0x40000000 with shmid 655389.
Allocated segment at address 0xa0000000 with shmid 1835020.
Allocated segment at address 0x80000000 with shmid 786450.
Allocated segment at address 0x90000000 with shmid 1179663.
db2pd> -memp


Database Partition 0 -- Active -- Up 0 days 03:02:20


Memory Pools:
Address    MemSet   PoolName   Id    Overhead   LogSz      LogUpBnd
0x30000840 DBMS     fcm        74    16240      417792     450560
0x300007A0 DBMS     monh       11    24192      142032     368640
0x30000700 DBMS     resynch    62    12112      102504     3244032
0x30000660 DBMS     ostrack    57    13184      5000400    5046272
0x300005C0 DBMS     apmh       70    0          38730      2129920
0x30000520 DBMS     kerh       52    32         31488      376832
0x30000480 DBMS     bsuh       71    3360       3598864    17137664
0x300003E0 DBMS     sqlch      50    0          762400     786432
0x30000340 DBMS     pmth       80    0          72         216540
0x300002A0 DBMS     krcbh      69    0          19228      32768
0x30000200 DBMS     debug      87    16240      12353536   12353552
0x80000700 FCM      fcmrqb     79    16240      835632     1548336
0x80000660 FCM      fcmce      78    16240      225328     696368
0x800005C0 FCM      fcma       75    16240      237616     712752
0x80000520 FCM      fcmrqb     79    16240      835632     1548336
0x80000480 FCM      fcmce      78    16240      225328     696368
0x800003E0 FCM      fcma       75    16240      237616     712752
0x80000340 FCM      fcmbp      13    404848     17011260   17980332
0x800002A0 FCM      debug      87    16240      12353536   12517392
0x80000200 FCM      eduah      72    5984       256024     419888
0x900002A0 FMP      undefh     59    8032       122900     245163840
0x90000200 FMP      debug      87    16240      12353536   12517392

db2pd> -age


LogHWM     PhySz      PhyUpBnd   PhyHWM     Bnd BlkCnt CfgParm
417792     458752     458752     458752     Ovf 4      n/a
142116     180224     376832     180224     Ovf 13     MON_HEAP_SZ
102504     131072     3244032    131072     Ovf 2      n/a
5000400    5029888    5046272    5029888    Ovf 6      n/a
39010      49152      2129920    49152      Ovf 21     n/a
```

```
7    32396      49152      376832     49152      Ovf 11     n/a
7    3605820    3620864    17137664   3620864    Ovf 55     n/a
7    762400     786432     786432     786432     Ovf 163    n/a
7    252        16384      229376     16384      Ovf 2      n/a
7    19228      32768      32768      32768      Ovf 3      n/a
7    12353536   12369920   12369920   12369920   Ovf 1      n/a
7    835632     917504     1556480    917504     Phy 19     n/a
7    225328     294912     704512     294912     Phy 15     n/a
7    237616     294912     720896     294912     Phy 15     n/a
7    835632     917504     1556480    917504     Phy 19     n/a
7    225328     294912     704512     294912     Phy 15     n/a
7    237616     294912     720896     294912     Phy 15     n/a
7    17011260   17448960   17989632   17448960   Phy 38     n/a
7    12353536   12369920   12533760   12369920   Phy 1      n/a
7    256024     262144     425984     262144     Phy 1      n/a
7    122900     131072     245170176  131072     Phy 1      n/a
7    12353536   12369920   12533760   12369920   Phy 1      n/a
7
7    Database Partition 0 -- Active -- Up 0 days 03:02:23
7
7
7    Agents:
7    Current agents:     8
7    Idle agents:        6
7    Active agents:      2
7    Coordinator agents: 1
7
7
7    Address    AppHandl [nod-index] AgentPid   Priority   Type     State   ClientPid   LkTmOt DBName
7    0x315898C0 0        [000-00000] 155688     0                   Idle    n/a         0      n/a
7    0x31588880 0        [000-00000] 1204334    0                   Idle    n/a         0      n/a
7    0x314DE190 0        [000-00000] 700478     0                   Idle    n/a         0      n/a
7    0x315890A0 0        [000-00000] 643262     0                   Idle    n/a         0      n/a
7    0x31588060 0        [000-00000] 1134718    0                   Idle    n/a         0      n/a
7    0x314DD730 0        [000-00000] 676026     0          Panic    Idle    n/a         0      n/a
7    0x314DF1D0 55       [000-00055] 614516     0          SubAgent Active  794820      NotSet SAMPLE
7    0x314DE9B0 55       [000-00055] 1093854    0          Coord    Active  794820      NotSet SAMPLE
7    db2pd> q
7
7      Userid   ClientNm Rowsread   Rowswrtn
7      n/a      n/a      0          0
7      n/a      n/a      0          0
7      n/a      n/a      0          0
7      n/a      n/a      0          0
7      n/a      n/a      0          0
7      n/a      n/a      0          0
7      jmcmahon db2bp    40         0
7      jmcmahon db2bp    12         0
7
7    Removed segment with shmid 786451.
7    Removed segment with shmid 655389.
7    Removed segment with shmid 1835020.
7    Removed segment with shmid 786450.
7    Removed segment with shmid 1179663.
```

7 **Scenario 7**: Figuring out which application is using up your tablespace.

7 Using db2pd -tcbstats, the number of Inserts can be identified for a table. Here is
7 temp table TEMP1.

```
7    TCB Table Stats:
7    Address            TbspaceID TableID TableName       SchemaNm Scans    ObjClass UDI
7    0x000000022094AA58 4         2       TEMP1           SESSION  0        Temp     0
7
7    DataSize   IndexSize  PgReorgs   NoChgUpdts Reads                FscrUpdates
7    1          0          0          0          0                    0
```

```

Inserts         Updates       Deletes       OvFlReads       OvFlCrtes  LfSize    LobSize
21586           0             0             0               0          0          0
```

which can be mapped to tablespace 4 in db2pd -tablespaces output.

```
Tablespaces:
Address            Id    Type Content AS  AR  PageSize   ExtentSize Auto Prefetch
0x0000000220942F80 4     DMS  UsrTmp  No  No  4096       32         Yes  32

BufID      BufIDDisk  State      TotPages   UsablePgs  UsedPgs    PndFreePgs
1          1          0x00000000 10000      9952       128        0

FreePgs    InitSize    IncSize     IIP MaxSize     LastResize
9824       0           0           No  0           None

LRF HWM        MinRecTime NQuiescers FSC NumCntrs   MaxStripe  Name
No  128        0          0          On  1          0          TEMPSPACE2

Containers:
Address            TspId ContainNum Type  TotalPages UseablePgs
0x0000000220377CE0 4     0          File  10000      9952

StripeSet  Container
0          /export/home/jmcmahon/tempspace2a
```

(The UsablePgs vs. TotalPages is where they would notice the space filling)

Once this is known, we can identify the dynamic sql statement using a table called "TEMP1."

```
db2pd -db sample -dyn

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:13:06

Dynamic Cache:
Current Memory Used           1022197
Total Heap Size               1271398
Cache Overflow Flag           0
Number of References          237
Number of Statement Inserts   32
Number of Statement Deletes   13
Number of Variation Inserts   21
Number of Statements          19

Dynamic SQL Statements:
Address            AnchID StmtUID  NumEnv  NumVar  NumRef  NumExe  Text
0x0000000220A08C40 78     1        2       2       3       2       declare global temporary
    table temp1 (c1 char(6)) not logged
0x0000000220A8D960 253    1      1      1     24       24       insert into session.temp1
    values('TEST')
```

And finally, map this to -app output to identify the application.

```
Applications:
Address            AppHandl [nod-index] NumAgents  CoorPid    Status
0x0000000200661840 501      [000-00501] 1          11246      UOW-Waiting

C-AnchID C-StmtUID  L-AnchID L-StmtUID  Appid
0        0          253      1          *LOCAL.jmcmahon.050202160426

Also, the -agent output will show the number of row written as verification.

Address            AppHandl [nod-index] AgentPid   Priority   Type       DBName
0x0000000200698080 501      [000-00501] 11246      0          Coord      SAMPLE
```

```
State       ClientPid  Userid   ClientNm Rowsread   Rowswrtn   LkTmOt
Inst-Active 26377      jmcmahon db2bp    22         9588       NotSet
```

Scenario 8: Monitoring recovery. db2pd -recovery shows several counters to make sure recovery is progressing. Current Log and Current LSN provide the log position. CompletedWork counts the number of bytes completed thus far.

```
Recovery:
Recovery Status     0x00000401
Current Log         S0000005.LOG
Current LSN         000002551BEA
Job Type            ROLLFORWARD RECOVERY
Job ID              7
Job Start Time      (1107380474) Wed Feb  2 16:41:14 2005
Job Description     Database Rollforward Recovery
Invoker Type        User
Total Phases        2
Current Phase       1

Progress:
Address            PhaseNum   Description         StartTime                 CompletedWork
0x0000000200667160 1          Forward             Wed Feb  2 16:41:14 2005   2268098 bytes
0x0000000200667258 2          Backward            NotStarted                0 bytes
```

Scenario 9: Understanding how much resource a transaction is using. db2pd -transactions provides the number of locks, first lsn, last lsn, logspace used, and space reserved. This can be useful for understanding the behavior of any transaction.

```
Transactions:
Address            AppHandl [nod-index] TranHdl    Locks      State   Tflag
0x000000022026D980 797      [000-00797] 2          108        WRITE   0x00000000
0x000000022026E600 806      [000-00806] 3          157        WRITE   0x00000000
0x000000022026F280 807      [000-00807] 4          90         WRITE   0x00000000

Tflag2     Firstlsn       Lastlsn        LogSpace       SpaceReserved
0x00000000 0x000001072262 0x0000010B2C8C 4518           95450
0x00000000 0x000001057574 0x0000010B3340 6576           139670
0x00000000 0x00000107CF0C 0x0000010B2FDE 3762           79266

TID            AxRegCnt   GXID
0x000000000451 1          0
0x0000000003E0 1          0
0x000000000472 1          0
```

Scenario 10: Monitoring log usage. db2pd -logs is useful to monitor log usage for a database. Watching the Pages Written output tells whether log usage is progressing or not.

```
Logs:
Current Log Number           4
Pages Written                464

Address            StartLSN       State      Size       Pages      Filename
0x000000022022FEB8 0x000000FA0000 0x00000000 1000       597        S0000000.LOG
0x000000022022FF78 0x000001388000 0x00000000 1000       5          S0000001.LOG
0x0000000220008E78 0x000001770000 0x00000000 1000       3          S0000002.LOG
0x0000000220A57F58 0x000001B58000 0x00000000 1000       1000       S0000003.LOG
0x0000000220A32598 0x000001F40000 0x00000000 1000       1000       S0000004.LOG
```

For Saturn, db2pd -logs has some new information:

```
Logs:
Current Log Number           2
Pages Written                846
Method 1 Archive Status      Success
Method 1 Next Log to Archive  2
Method 1 First Failure        n/a
Method 2 Archive Status       Success
Method 2 Next Log to Archive  2
Method 2 First Failure        n/a

Address            StartLSN       State      Size       Pages      Filename
0x000000023001BF58 0x000001B58000 0x00000000 1000       1000       S0000002.LOG
0x000000023001BE98 0x000001F40000 0x00000000 1000       1000       S0000003.LOG
0x0000000230008F58 0x000002328000 0x00000000 1000       1000       S0000004.LOG
```

Two problems can be identified with this output.

1. If there is a problem with archiving, Archive Status will be set to Failure indicating the most recent log archive failed. Or if there is an ongoing archive failure preventing logs from archiving at all First Failure will be set.
2. If log archiving is proceeding very slowly, Next Log to Archive will be behind Current Log Number. This can cause the log path to fill up, which in turn, can prevent any data changes from occurring in the database when the log path is completely filled.

Scenario 11: Sysplex list. Without `db2pd -sysplex`, the only way to report the sysplex list is with db2trc.

```
Sysplex List:
Alias:         LAKETCP
Location Name: LAKEERIE
Count:         1

IP Address      Port       Priority   Connections Status     PRDID
   9.26.89.87      448        1          0           0
```

db2support - Collecting environment information

When it comes to collecting information for a DB2 problem, the most important DB2 utility you need to run is db2support.

The db2support utility is designed to automatically collect all DB2 and system diagnostic information available. It also has an optional interactive ″Question and Answer″ session, which poses questions about the circumstances of your problem.

Using db2support avoids possible user errors, as you don't need to manually type commands such as ″`GET DATABASE CONFIGURATION FOR <database name>`″ or ″`LIST TABLESPACES SHOW DETAIL`″. Also, you don't require instructions on which commands to run or what files to collect, therefore information-gathering for problem determination is quicker.

Note: The db2support utility should be run by a user with SYSADM authority, such as an instance owner, so that the utility can collect all of the necessary information without an error. If a user without SYSADM authority runs db2support, SQL errors (SQL1092) might result when the utility runs commands such as ″query client″ or ″list active databases″.

If you're using db2support to help convey information to IBM software support, run db2support while the system is experiencing the problem that led you to request support. That way, the tool will collect operating system performance information. You can also run db2support after the fact for problems requiring post-mortem analysis (for example, traps or crashes).

Executing `db2support -h` brings up the complete list of possible options you can run the utility with. The following basic invocation is usually sufficient for collecting most of the information required to debug a problem (note that if the -c option is used the utility will establish a connection to the database):

```
db2support <output path> -d <database name> -c
```

The output is conveniently collected and stored in a compressed ZIP archive, `db2support.zip`, so that it can be transferred and extracted easily on any system.

The db2support tool adds the output of the various commands that it runs to the archive as files without encapsulating them in a directory (as of DB2 v.7, FixPak 7). Previously, generated command output was stored in either db2support.html or detailed_system_info.html. db2support no longer store all it's output files under one output directory. Rather, db2support creates multiple subdirectories, and organizes each output file under an appropriate subdirectory. This avoids a mass of files existing under a single directory, and having to search through the list of files to find the one of interest. The subdirectory names help organize and categorize the type of data that is captured. For example, the database manager configuration, which has the output filename of `dbm.supp_cfg`, is stored under the subdirectory named `DB2CONFIG`. The diagnostic files and contents that exist under `DIAGPATH` (e.g., `db2diag.log`) is stored under the subdirectory named `DB2DUMP`.

The type of information that db2support captures depends on the way the command is invoked, whether or not the database manager has been started, and whether it's possible to connect to the database.

The db2support utility collects the following information under all conditions:

- db2diag.log
- All trap files
- Lock list files
- Dump files
- Buffer pool and table space (SQLSPCS.1 and SQLSPCS.2) control files (with -d)
- Various system related files
- Output from various system commands
- db config (with -d)
- dbm config files
- Log File Header file (with -d)
- Recovery History File (with -d)
- db2cli.ini

Depending on the circumstances, db2support may also collect:

- Active log files
- Contents of the db2dump directory (in other words, what was not collected above)
- Core files (-a for all core files, -r for only the most recent core file)
- Extended system information (-s)

The HTML report db2support.html will always include the following information:

- PMR number, if one exists (if -n was specified)
- Operating system and level (for example, AIX 4.5)
- DB2 release information
- Engine library header information
- 32- vs 64-bit environment

- DB2 install path information
- Contents of db2nodes.cfg (for DB2 UDB Enterprise Server Edition)
- Number of CPUs and disks and how much memory
- List of databases on this instance
- Registry information and environment, including path and libpath
- Disk freespace for current filesystem and inodes for UNIX
- JDK level
- dbm config
- Listing of the database recovery history file
- ’ls -lR’ (or Windows equivalent) of the sqllib directory
- The result of the LIST NODE DIRECTORY command
- The result of the LIST ADMIN NODE DIRECTORY command
- The result of the LIST DCS DIRECTORY command
- The result of the LIST DCS APPLICATIONS EXTENDED command
- List of all installed software

The following information appears in the db2support.html file when -s is specified:

- Detailed disk information (partition layout, type, LVM information, and so on)
- Detailed network information
- Kernel statistics
- Firmware versions
- Other platform specific commands

The db2support.html file contains the following information if DB2 has been started:

- Client connection state
- db/dbm config (db cfg require -d option)
- CLI config
- Memory pool info (size and consumed). Complete data if -d option used
- The result of the LIST ACTIVE DATABASES command
- The result of the LIST DATALINKS MANAGERS command
- The result of the LIST DCS APPLICATIONS command

The db2support.html file contains the following information if -c has been specified and a connection to the database can be made

- Number of user tables
- Approximate size of DB data
- Database snapshot
- Application snapshot
- Buffer pool information
- The result of the LIST APPLICATIONS command
- The result of the LIST COMMAND OPTIONS command
- The result of the LIST DATABASE DIRECTORY command
- The result of the LIST INDOUBT TRANSACTIONS command
- The result of the LIST NODEGROUPS command
- The result of the LIST NODES command

- The result of the LIST ODBC DATA SOURCES command
- The result of the LIST PACKAGES/TABLES command
- The result of the LIST TABLESPACE CONTAINERS command
- The result of the LIST TABLESPACES command
- The result of the LIST DRDA IN DOUBT TRANSACTIONS command

Related reference:

- db2support - Problem Analysis and Environment Collection Tool Command

Platform specific tools

Diagnostic tools (Windows)

The following diagnostic tools are available for Windows NT, Windows 2000 and Windows XP operating systems:

Event log, performance monitor, and other administrative tools
The Administrative Tools folder provides a variety of diagnostic information, including access to the event log and access to performance information.

Task Manager
The Task Manager shows all of the processes running on the Windows server, along with details about memory usage. Use this tool to find out which DB2 processes are running, and to diagnose performance problems. Using this tool, you can determine memory usage, memory limits, swapper space used, and memory leakage for a process.

To open the Task Manager, press `<Ctrl> + <Alt> + <Delete>`, and click **Task Manager** from the available options.

Dr. Watson
The Dr. Watson utility is invoked in the event of a General Protection Fault (GPF). It logs data that may help in diagnosing a problem, and saves this information to a file. You must start this utility by typing `drwatson` on the command line.

DB2-supplied tools
DB2 supplies administration and development tools to help you identify DB2 problems such as the administration notification logs.

ODBC and CLI traces
CLI traces help to identify problems in CLI and ODBC applications.

SNA server tracing
If you have the SNA server installed, it provides a tracing facility. To use it, go to the SNA Server window.

Related concepts:

- on page 0
- "Diagnostic tools (UNIX)" on page 38

Diagnostic tools (UNIX)

This section describes some essential UNIX-based commands for troubleshooting and performance monitoring. For details on any one of these commands, precede it with man on the command line. Use these commands to gather and process data that can help identify the cause of a problem you are having with your

UNIX®-based system. Once the data is collected, it can be examined by someone who is familiar with the problem, or provided to DB2® UDB customer support if requested.

Troubleshooting commands (AIX): The following AIX system commands are useful for DB2 troubleshooting:

errpt

The `errpt` command reports system errors such as hardware errors and network failures.

- For an overview that shows one line per error, use `errpt`
- For a more detailed view that shows one page for each error, use `errpt -a`
- For errors with an error number of ″1581762B″, use `errpt -a -j 1581762B`
- To find out if you ran out of paging space in the past, use `errpt | grep SYSVMM`
- To find out if there are token ring card or disk problems, check the errpt output for the phrases ″disk″ and ″tr0″

lsps

The `lsps -a` command monitors and displays how paging space is being used.

lsattr

This command displays various operating system parameters. For example, use the following command to find out the amount of real memory on the node:

```
lsattr -l sys0 -E
```

This example lets you see the maximum number of processes per user.

xmperf

For AIX systems using Motif, this command starts a graphical monitor that collects and displays system-related performance data. The monitor displays 3-dimensional diagrams for each node in a single window, and is good for high-level monitoring. However, if activity is low, the output from this monitor is of limited value.

Troubleshooting commands (UNIX): The following UNIX-based system commands are also useful for DB2 troubleshooting. These commands are for all UNIX-based systems, including AIX, unless otherwise noted.

df

The `df` command lets you see if file systems are full.

- To see how much free space is in all file systems (including mounted ones), use `df`
- To see how much free space is in all file systems with names containing ″dev″, use `df | grep dev`
- To see how much free space is in your home file system, use `df /home`
- To see how much free space is in the file system ″tmp″, use `df /tmp`
- To see if there is enough free space on the machine, check the output from the following commands: `df /usr , df /var , df /tmp , and df /home`

truss

Available for SVR4 UNIX-based environments such as Solaris Operating Environment, and Silicon Graphics IRIX, this command is useful for tracing system calls in one or more processes. The `truss` command is not available for AIX.

pstack

Available for Solaris 2.5.1 or later, the `/usr/proc/bin/pstack` command displays stack traceback information. The `/usr/proc/bin` directory contains other tools for debugging processes that appear to be suspended.

Performance Monitoring Tools:

The following tools are available for monitoring the performance of your UNIX-based system.

vmstat

This command is ideal for monitoring paging rate, which can be found under the page in (pi) and page out (po) columns. Other important columns are the amount of allocated virtual storage (avm) and free virtual storage (fre).

This command is useful for determining if something is suspended or just taking a long time.

iostat

This command is useful for monitoring I/O activities. You can use the read and write rate to estimate the amount of time required for certain SQL operations (if they are the only activity on the system).

This command is also useful for determining if something is suspended or just taking a long time.

netstat

This command lets you know the network traffic on each node, and the number of error packets encountered. It is useful for isolating network problems.

system file

Available for Solaris Operating Environment, the `/etc/system` file contains definitions for kernel configuration limits such as the maximum number of users allowed on the system at a time, the maximum number of processes per user, and the inter-process communication (IPC) limits on size and number of resources. These limits are important because they affect DB2 performance on a Solaris Operating Environment machine. See the Quick Beginnings information for further details.

Commands for DB2 in a partitioned database environment (UNIX): For multi-partitioned systems, you can run UNIX-based commands on all database partitions by surrounding the commands with quotation marks and preceding them with one of the following prefix commands:

db2_all

Provides information for all logical nodes. The following example shows active applications for all logical nodes:

```
db2_all ";db2 LIST APPLICATIONS"
```

The semicolon (;) is optional and it improves performance by issuing commands simultaneously to all nodes.

rah

Provides information for all physical nodes. This command is useful for filtering out multiple entries that might occur using db2_all when there are multiple logical nodes on machines. The following example lists the first three lines of the hardware error log for each physical node:

```
rah ";errpt | head -3"
```

The semicolon (;) is optional and it improves performance by issuing commands simultaneously to all nodes.

The following commands are also useful for multi-partitioned systems:

spmon

If using multiple nodes on RS/6000 SP systems, you may need to check if the high performance switch (HPS) is running on all workstations.

To view the status of all nodes, use one of the following commands from the control workstation:

- `spmon -d` for ASCII output
- `spmon -g` for a graphical user interface

Alternatively, use the command `netstat -i` from a node workstation to see if the switch is down. If the switch is down, there is an asterisk (*) beside the node name. For example:

```
css0* 65520 <Link>0.0.0.0.0.0
```

The asterisk does not appear if the switch is up.

Related reference:

- Diagnostic tools for Windows operating systems
- Administration logs, error logs and first failure data capture

Determining active process status

There are commands that can be run either within DB2 or at the operating system level, in order to determine the status of various DB2 processes. Several of these are demonstrated here. Use these commands to look for discrepancies between DB2(R) processes that are running and DB2 processes that you expect to be there.

The db2_local_ps command (Linux/UNIX):

On Linux/UNIX platforms, all of the DB2 processes running under an instance can be displayed using the `db2_local_ps` command:

```
/home/db2v8> db2_local_ps

(/home/db2v8) $ db2_local_ps
```

```
Node 1
     UID       PID       PPID    C     STIME     TTY     TIME CMD
   db2v8     41292      40012    0       Jan      19        - 0:00 db2sysc 1
   db2v8     28270      41292    0       Jan      19        - 6:34 db2fcmdm 1
   db2v8     37768      41292    0       Jan      19        - 0:00 db2gds 1
   db2v8     38928      41292    0       Jan      19        - 0:00 db2panic (idle) 1
   db2v8     39578      41292    0       Jan      19        - 0:00 db2tcpcm 1
   db2v8     40284      41292    0       Jan      19        - 0:00 db2resync 1
   db2v8     40910      41292    0       Jan      19        - 0:00 db2ipccm 1
    root     41030      41292    0       Jan      19        - 0:00 db2ckpwd 1
   db2v8     42058      41292    0       Jan      19        - 0:00 db2pdbc 1
    root     43348      41292    0       Jan      19        - 0:00 db2ckpwd 1
    root     43864      41292    0       Jan      19        - 0:00 db2ckpwd 1
   db2v8     44380      41292    0       Jan      19        - 0:00 db2spmrsy 1
   db2v8     33514      37768    0       Jan      19        - 0:00 db2srvlst 1
   db2v8     44896      37768    0       Jan      19        - 0:00 db2spmlw 1
   db2v8     45532      37768    0  10:12:05       -     0:03 db2cart 1
   db2v8     45738      40910    0  10:12:04       -     0:00 db2agent (instance) 1
   db2v8     48518      40910    0  10:13:50       -     1:20 db2agent (idle) 1

Node 2 ...
```

Note that no processes will be shown if the instance is stopped. Run the db2start command if no processes are listed.

The db2stat command (Windows):

On Windows platforms, all of the DB2 processes running under an instance can be displayed using the `db2stat` command:

```
C:\>db2stat

Environment Strings
 --> DB2INSTANCE=DB2
 --> DB2TEMPDIR=D:\IBM\SQLLIB\

DB2 Processes
        DB2JDS          960    x3C0
        DB2LICD         1004   x3EC
        DB2SEC          1020   x3FC
        DB2RCMD         1036   x40C
        DB2SYSTRAY      1508   x5E4
        DB2SYSCS        1780   x6F4
        DB2FMP          1424   x590
        DB2STAT         1988   x7C4
```

One thing to note in the Windows case is that because DB2 is thread-based (not process-based), you will only see one process (DB2SYSCS) for all of an instance's EDUs. It is obvious that the same degree of information is not returned in Windows as is returned in UNIX, but it is still useful at times to know the process ID for a running instance. For example, you can use the Task Manager utility to determine the CPU and memory usage for a given process ID.

:

The **ps** command is a UNIX®-based system command that returns process status information about active processes (to standard output). Use it to look for discrepancies between DB2® processes that are running and DB2 processes that you expect to be there.

Flags control the types of information displayed for each active process, and may be applied simultaneously to give a cumulative effect. You can access information on the command syntax and flag options by using the **man ps** command on a system command prompt.

Example: To show all processes of the instance ID "svtdbm" use: `ps -fu svtdbm`

The following sample shows typical output from this command. Additional processes would appear on multi-partitioned systems.

```
 svtdbm 1  5112 2 27894 3  0 10:48:17 4     -  0:00 db2ipccm 5
svtdbm   6135 27894   0 10:48:17      -  0:00 db2gds
svtdbm  14329 27894   0 10:48:17      -  0:00 db2resyn
svtdbm  15356 27894   0 10:48:17      -  0:00 db2snacm 6
svtdbm  18682 27894   0 10:48:17      -  0:00 db2tcpcm 6
svtdbm  27894 31989   0 10:48:17      -  0:00 db2sysc 7
svtdbm  33275 27894   0 10:48:17      -  0:00 db2tcpim 6
svtdbm  39939 18682   7 10:48:19      -  0:00 db2agent
svtdbm  56074 34761   2 10:48:37 pts/10  0:00 db2
```

Legend:

1. The instance ID “(return to example)”
2. The process identifier (pid) “(return to example)”
3. The parent process identifier “(return to example)”
4. The timestamp “(return to example)”
5. The name of the process. The related links for more information. “(return to example)”
6. The communication listeners (in this sample, APPC and TCP/IP listeners and the TCP/IP interrupt manager) are up “(return to example)”
7. The system controller process

   On UNIX-based systems other than AIX® and SCO OpenServer, the db2sysc process is the only process shown for all server-side processes (for example, agents, loggers, page cleaners, and prefetchers). On Solaris systems, you can see these side processes with the command **/usr/ucb/ps axw**. “(return to example)”

Related reference:

- Understanding the DB2 process model

Trace files

Basic trace diagnostics

If you experience a recurring and reproducible problem with DB2, tracing sometimes allows you to capture additional information about it. Under normal circumstances, you should only use a trace if asked to by DB2 Customer Support. The process of taking a trace entails setting up the trace facility, reproducing the error and collecting the data.

The amount of information gathered by a trace grows rapidly. When you take the trace, capture only the error situation and avoid any other activities whenever possible. When taking a trace, use the smallest scenario possible to reproduce a problem.

The process of performing traces often has a global effect on the behavior of a DB2 instance. The degree of performance degradation is dependent on the type of problem and on how many resources are being used to gather the trace information.

DB2 Customer Support should provide the following information when traces are requested:

- Simple, step by step procedures
- An explanation of where each trace is to be taken
- An explanation of what should be traced
- An explanation of why the trace is requested
- Backout procedures (i.e. how to disable all traces)

Though you should be counseled by DB2 Customer Support as to which traces to obtain, here are some general guidelines as to when you'd be asked to obtain particular traces:

- If the problem occurs during installation, and the default installation logs are not sufficient to determine the cause of the problem, installation traces are appropriate. See **Tracing installation problems**
- If the problem occurs in one of the GUI (Graphical User Interface) tools, and the same actions succeed when performed via explicit commands in the DB2 command window, then a Control Center trace is appropriate. Note that this will only capture problems with tools that can be launched from the Control Center.
- If the problem manifests in a CLI application, and the problem cannot be recreated outside of the application, then a CLI trace is appropriate.
- If the problem manifests in a JDBC application, and the problem cannot be recreated outside of the application, then a JDBC trace is appropriate.
- If the problem is directly related to information that is being communicated at the DRDA layer, a DRDA trace is appropriate.
- For all other situations where a trace is feasible, a DB2 trace is most likely to be appropriate.

Trace information is not always helpful in diagnosing an error. For example, it may not capture the error condition in the following situations:

- The trace buffer size you specified was not large enough to hold a complete set of trace events, and useful information was lost when the trace stopped writing to the file or wrapped.
- The traced scenario did not re-create the error situation.
- The error situation was re-created, but the assumption as to where the problem occurred was incorrect. For example, the trace was collected at a client workstation while the actual error occurred on a server.

DB2 trace files

The **db2trc** command controls the trace facility provided with DB2. The trace facility records information about operations and formats this information into a readable form.

Please keep in mind that there is added overhead when a trace is running so enabling the trace facility may impact your system's performance.

In general, DB2 Support and development teams use DB2 traces to debug customer problems. You might run a trace to gain further information about a problem that you are investigating, but its use is rather limited without knowledge of the DB2 source code.

Nonetheless, it is important to know how to correctly turn on tracing and how to dump trace files, just in case you need to send one to the DB2 support and development teams.

Note: You will need one of SYSADM, SYSCTRL or SYSMAINT authority to use `db2trc`

Command syntax:

To get a general idea of the options available, execute the db2trc command without any parameters:

```
C:\>db2trc Usage: db2trc (chg|clr|dmp|flw|fmt|inf|off|on) options
```

Command parameters:

chg|change
Change the trace mask, maxSysErrors or maxRecordSize

clr|clear
Clear the trace

dmp|dump
Dump the trace to a binary trace file

flw|flow
Show control flow of the trace

fmt|format
Format the trace

inf|info|information
Get information on the trace off Turn the trace off on Turn the trace on

For more information about a specific db2trc command parameter, use the `-u` option. For example, to see more information about turning the trace on, execute the following command:

```
db2trc on -u
```

This will provide information about all of the additional options (labeled as ″facilities″) that can be specified when turning on a DB2 trace.

The most important option that you need to be aware for turning on trace is `-L`. This specifies the size of the memory buffer that will be used to store the information being traced. The buffer size can be specified in either bytes or megabytes. (To specify megabytes append either ″`M`″ or ″`m`″ after the value). The trace buffer size must be a power of two megabytes. If you specify a size that does not meet this requirement, the buffer size will automatically be rounded down to the nearest power of two.

If the buffer is too small, information might be lost. By default only the most recent trace information is kept if the buffer becomes full. If the buffer is too large, it might be difficult to send the file to the DB2 support team.

If tracing an operation that is relatively short (such as a database connection), a size of approximately 8MB might be sufficient:

```
C:\> db2trc on -l 8M Trace is turned on
```

However, if you are tracing a larger operation or if a lot of work is going on at the same time, a larger trace buffer may be required.

On most platforms, tracing can be turned on at any time and works as described above. However, there are certain situations to be aware of:

1. On multiple database partition systems, you must run a trace for each physical (as opposed to logical) database partition.
2. On Solaris platforms, if the trace is turned off after the instance has been started, a very small buffer will be used regardless of the size specified. To effectively run a trace on Solaris, turn the trace on before starting the instance and ″clear″ it as necessary afterwards.

DB2 trace memory buffers: The most important option that you need to be aware for turning on trace is -l. This specifies the size of the memory buffer that will be used to store the information being traced. The buffer size can be specified in either bytes or megabytes. (To specify megabytes append either ″M″ or ″m″ after the value). The trace buffer size must be a power of two megabytes. If you specify a size that does not meet this requirement, the buffer size will automatically be rounded down to the nearest power of two.

If the buffer is too small, information might be lost. By default only the most recent trace information is kept if the buffer becomes full. If the buffer is too large, it might be difficult to send the file to the DB2 support team.

If tracing an operation that is relatively short (such as a database connection), a size of approximately 8MB might be sufficient:

```
C:\> db2trc on -l 8M
Trace is turned on
```

However, if you are tracing a larger operation or if a lot of work is going on at the same time, a larger trace buffer may be required.

On most platforms, tracing can be turned on at any time and works as described above. However, there are certain situations to be aware of:

1. On multiple database partition systems, you must run a trace for each physical (as opposed to logical) database partition.
2. On Solaris platforms, if the trace is turned on after the instance has been started, a very small buffer will be used regardless of the size specified. To effectively run a trace on Solaris, turn the trace on before starting the instance and ″clear″ it as necessary afterwards.

Dumping a trace file: Once the trace facility has been enabled using the on option, all subsequent work done by the instance will be traced.

While the trace is running, you can use the clr option to clear out the trace buffer. All existing information in the trace buffer will be removed.

```
C:\>db2trc clr
Trace has been cleared
```

Once the operation being traced has finished, use the dmp option followed by a trace file name to dump the memory buffer to disk. For example:

```
C:\>db2trc dmp trace.dmp
Trace has been dumped to file
```

The trace facility will continue to run after dumping the trace buffer to disk. To turn tracing off, use the off option:

```
C:\>db2trc off
Trace is turned off
```

Formatting a DB2 trace file: The dump file created above is in a binary format that is not readable.

To verify that a trace file can be read, format the binary trace file to show the flow control and send the formatted output to a null device. The following example shows the command to perform this task:

```
db2trc flw example.trc nul
```

The output for this command will explicitly tell you if there is a problem reading the file, and whether or not the trace was wrapped.

At this point, the dump file could be sent to DB2 Support. They would then format it based on your DB2 service level. However, you may sometimes be asked to format the dump file into ASCII format before sending it. This is accomplished via the `flw` and `fmt` options. You must provide the binary dump file along with the name of the ASCII file that you want to create:

```
C:\>db2trc flw trace.dmp trace.flw
C:\Temp>db2trc flw trace.dmp trace.flw
Total number of trace records     : 18854
Trace truncated                   : NO
Trace wrapped                     : NO
Number of trace records formatted : 1513 (pid: 2196 tid 2148 node: -1)
Number of trace records formatted : 100 (pid: 1568 tid 1304 node: 0)
...

C:\>db2trc fmt trace.dmp trace.fmt
C:\Temp>db2trc fmt trace.dmp trace.fmt
Trace truncated                   : NO
Trace wrapped                     : NO
Total number of trace records     : 18854
Number of trace records formatted : 18854
```

If this output indicates ″Trace wrapped″ is ″YES″, then this means that the trace buffer was not large enough to contain all of the information collected during the trace period. A wrapped trace might be okay depending on the situation. If you are interested in the most recent information (this is the default information that is maintained, unless the -L option was used to specify otherwise), then what is in the trace file might be sufficient. However, if you are interested in what happened at the beginning of the trace period or if you are interested in everything that occurred, you might want to redo the operation with a larger trace buffer.

Another thing to be aware of is that on UNIX platforms, DB2 will automatically dump the trace buffer to disk when it shuts the instance down due to a severe error. Thus if tracing is enabled when an instance ends abnormally, a file will be created in the diagnostic directory and its name will be db2trdmp.###, where ### is the node number. This does not occur on Windows platforms. You have to dump the trace manually in those situations.

To summarize, the following is an example of a common use of db2trc:

```
db2trc on -l 8M
db2trc clr
<execute command>
db2trc dump db2trc.dmp
db2trc off
db2trc flw db2trc.dmp <filename>.flw
db2trc fmt db2trc.dmp <filename>.fmt
db2trc fmt -c db2trc.dmp .fmtc
```

DRDA trace files

Before analyzing DRDA traces, you need to understand that DRDA is an open standard for definition of data and communication structures. i.e. DRDA comprises a set of rules about how data should be organized for transmission and how communication of that information should occur. These rules are defined in the following reference manuals:

- DRDA V3 Vol. 1: Distributed Relational Database Architecture
- DRDA V3 Vol. 2: Formatted Data Object Content Architecture
- DRDA V3 Vol. 3: Distributed Data Management Architecture

PDF versions of these manuals are available on www.opengroup.com.

The **db2drdat** utility records the data interchanged between the DB2 Connect server (on behalf of the database client) and the host or iSeries™ database server.

As a database administrator (or application developer), you may find it useful to understand how this flow of data works, because this knowledge can help you determine the origin of a particular problem. For example, if you issue a CONNECT TO database statement for a host or iSeries database server, but the command fails and you receive an unsuccessful return code. If you understand exactly what information was conveyed to the host or iSeries database server management system, you may be able to determine the cause of the failure even if the return code information is general. Many failures are caused by simple user errors.

Command syntax:

To get a general idea of the options available, issue the following command:

```
db2drdat -help
```

This will show the command syntax:

```
db2drdat on  [-i] [-c] [-r] [-s] [-l=LENGTH]
db2drdat off [-t=TRACEFILE] [-p=PROCESSID]
```

Command parameters:

on Turns the DRDA trace on

off Turns the DRDA trace off

-i Include timestamps

-c Trace SQLCAs

-r Trace DRDA receive buffers

-s Trace DRDA send buffers

-l=*LENGTH*
Sets the size of the trace buffer (in bytes). The default value is one megabyte.

-t=*TRACEFILE*
TRACEFILE is the file name where the captured trace will be stored. The default file name is db2drdat.dmp.

-p=*PROCESSID*
PROCESSID is the ID of a process to be traced. If PROCESSID is not specified, all process IDs for the DB2 instance are traced.

Usage:

The following is an example of a common use of db2drdat:

```
db2drdat on
<Execute command>
db2drdat off
<Collect the trace file db2drdat.dmp>
```

Output from `db2drdat` lists the data streams exchanged between the DB2 Connect workstation and the host or iSeries database server management system. Data sent to the host or iSeries database server is labeled SEND BUFFER and data received from the host or iSeries database server is labeled RECEIVE BUFFER.

If a receive buffer contains SQLCA information, it will be followed by a formatted interpretation of this data and labeled SQLCA. The SQLCODE field of an SQLCA is the *unmapped* value as returned by the host or iSeries database server. The send and receive buffers are arranged from the oldest to the most recent within the file. Each buffer has:

- The process ID
- A SEND BUFFER, RECEIVE BUFFER, or SQLCA label. The first DDM command or object in a buffer is labeled DSS TYPE.

The remaining data in send and receive buffers is divided into five columns, consisting of:

- A byte count.
- Columns 2 and 3 represent the DRDA(R) data stream exchanged between the two systems, in ASCII or EBCDIC.
- An ASCII representation of columns 2 and 3.
- An EBCDIC representation of columns 2 and 3.

Here is an example of a trace entry in a db2drda trace file:

```
2 data DB2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 102974 tid 1 cpid -1 node 0 probe 1177
 bytes 464

 SEND BUFFER(AR):

            EXCSAT RQSDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF 0123456789ABCDEF
  0000  00B4D041000100AE 1041006E115E8482   ...A.....A.n.^.. ..}........>.;db
  0010  F282974040404040 4040404040404040   ...@@@@@@@@@@@@@  2bp
  0020  4040F0F0F0F1F9F2 F3C5F0F0F0000000   @@.............. 0001923E000...
  0030  0000000000000000 0000000000000000   ................ ................
  0040  0000000000000000 000000000060F0F0   .............`.. .............-00
  0050  F0F1A2A495404040 4040404040404040   .....@@@@@@@@@@@  01sun
  0060  4040404040404040 4040404040404040   @@@@@@@@@@@@@@@@
  0070  C4C4C2F2C8D4E340 0018140414030007   .......@........  DDB2HMT........
  0080  2407000714740005 240F000714400007   $....t..$....@..  ...............
```

```
0090  000D1147D8C4C2F2 61F6F0F0F0000B11   ...G....a....... ....QDB2/6000...
00A0  6D99818398A485A3 000C115AE2D8D3F0   m.........Z.... _racquet...]SQL0
00B0  F8F0F2F0                            ....               8020
```

For more information, see the *DB2 for OS/390 Reference for Remote DRDA Requesters and Servers*, the *Distributed Relational Database Reference*, and the *Distributed Data Management Architecture Level 3: Reference*.

Related reference:

- db2drdat - DRDA Trace Command

JDBC trace files

There are two different methods for capturing JDBC traces, depending on the which type of JDBC driver your application uses.

Enabling JDBC traces: The JDBC trace is enabled by adding specific entries to the db2cli.ini file.

Note: There are lots of keywords that can be added to the db2cli.ini file that can affect application behavior. These keywords can resolve or be the cause of application problems. There are also some keywords that are not covered in the CLI documentation. Those are only available from DB2 Service and Support. If you have keywords in your db2cli.ini file that are not documented, it is likely that they were a recommendation from the DB2 support team.

By default, the location of the DB2™ CLI/ODBC configuration keyword file is in the sqllib directory on Window platforms, and in the sqllib/cfg directory of the database instance running the CLI/ODBC applications on UNIX™ platforms. If the ODBC Driver Manager is used to configure a User Data Source on the Windows™ platform, a db2cli.ini may be created in the user's home (profile) directory. The environment variable DB2CLIINIPATH can also be used to override the default and specify a different location for the file.

Step 1. Create a path for the trace files.

It is important to create a path that every user can write to. For example, on Windows:

```
mkdir c:\temp\trace
```

On UNIX:

```
mkdir /tmp/trace
chmod 777 /tmp/trace
```

Step 2. Update the CLI configuration keywords

This can be done by either (A) manually editing the db2cli.ini file or(B)using the UPDATE CLI CFG command.

Option A: Manually Editing the db2cli.ini file.

a. Open up the db2cli.ini file in a plain text editor.

b. Add the following section to the file (or if the COMMON section already exists, just append the variables):

```
[COMMON]
JDBCTrace=1
JDBCTracePathName=<path>
JDBCTraceFlush=1
```

where <path> is, for example, C:\temp\trace on Windows, or /tmp/trace on UNIX platforms.

c. Save the file with at least one blank line at the end of the file. (This prevents some parsing errors.)

Option B: Using UPDATE CLI CFG commands to update the db2cli.ini file. Issue the following commands:

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTrace 1
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTracePathName <path>
```

where <path> is, for example, C:\temp\trace on Windows, or /tmp/trace on UNIX platforms.

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTraceFlush 1
```

Step 3. Confirm the db2cli.ini configuration

Issue the following command to verify that the correct keywords are set and being picked up:

```
db2 GET CLI CFG FOR SECTION COMMON
```

Step 4. Restart the application

The db2cli.ini file is only read when the application starts, therefore, for any changes to take effect, the application must be restarted.

If tracing a JDBC stored procedure, this means restarting the DB2 instance.

Step 5. Capture the error

Run the application until the error is generated, then terminate the application. If it is possible, reduce the situation, such that the only JDBC applications that are running at the time of trace are those related to the problem recreation. This makes for much clearer trace files.

Step 6. Disable the JDBC trace

Set the JDBCTrace=0 keyword in the [COMMON] section of the db2cli.ini manually, or issue:

```
db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 0
db2 UPDATE CLI CFG FOR SECTION COMMON USING JDBCTrace 0
```

Restart any applications that are running and tracing.

Step 7. Collect the trace files

The JDBC trace files will be written to the path specified in the JDBCTracePathName keyword. The filenames generated will all end with a .trc extension. All files generated in the trace path at the time of the problem recreation are required.

When you use the trace facility to diagnose application issues, keep in mind that it does have an impact on application performance and that it affects all applications, not only your test application. This is why it is important to remember to turn it off after the problem has been identified.

Enabling DB2 Universal JDBC traces: Option A:

If you use the DataSource interface to connect to a data source, then use the DataSource.setTraceLevel() and DataSource.setTraceFile() method to enable tracing.

Option B:

If you use the DriverManager interface to connect to a data source, the easiest way to enable tracing will be to set the logWriter on DriverManager before obtaining a connection.

For example:

```
DriverManager.setLogWriter(new PrintWriter(new FileOutputStream("trace.txt")));
```

Option C:

If you are using the DriverManager interface, you can alternatively specify the traceFile and traceLevel properties as part of the URL when you load the driver. For example:

```
String databaseURL = "jdbc:db2://hal:50000/sample:traceFile=c:/temp/foobar.txt;" ;
```

More details on each of these options are provided in JDBC and SQLJ problem diagnosis with the DB2 Universal JDBC Driver.

Refer to one of the following:

- Obtaining traces of applications that use CLI-based Legacy Type 2 JDBC Driver, or
- Obtaining traces of applications that use the DB2 Universal JDBC Driver

Obtaining traces of applications that use CLI-based Legacy Type 2 JDBC Driver: The CLI-based Legacy Type 2 JDBC Driver for Linux, UNIX™, and Windows™ offers comprehensive tracing facilities. By default, these facilities are disabled and use no additional computing resources. When enabled, the trace facilities generate one or more text log files whenever an application accesses the driver). These log files provide detailed information about:

- the order in which JDBC functions were called by the application
- the contents of input and output parameters passed to and received from JDBC functions
- the return codes and any error or warning messages generated by JDBC functions

The CLI trace offers very little information about the internal workings of the DB2 CLI driver.

This type of trace is applicable for situations where a problem is encountered in:

- a JDBC application which uses the CLI-based Legacy Type 2 JDBC driver
- DB2 JDBC stored procedures[1]

Note: Internally, the CLI-based Legacy Type 2 JDBC Driver makes use of the DB2 CLI driver for database access. For example, the Java `getConnection()` method is internally mapped by the CLI-based Legacy Type 2 JDBC Driver to the DB2 CLI `SQLConnect()` function. As a result, Java developers might find a DB2 CLI trace to be a useful complement to the DB2 JDBC trace. See CLI Traces.

Related reference:

- CLI/ODBC/JDBC trace facility
- CLI and JDBC trace files

[1] Most of the DB2 CLI trace information and instructions which follow are generic and apply to both applications and stored procedures equally. However, unlike applications which are clients of a database server (and typically execute on a machine separate from the database server), stored procedures execute at the database server. Therefore, the following additional steps must be taken when tracing DB2 CLI stored procedures:

– Ensure the trace keyword options are specified in the db2cli.ini file located at the DB2 server.

– Ensure all keywords are configured correctly prior to database startup time (that is, when the db2start command is issued). Changing trace settings while the database server is running may have unpredictable results.

- db2cli.ini initialization file

Obtaining traces of applications that use the DB2 Universal JDBC Driver: To obtain data for diagnosing SQLJ or JDBC problems with the DB2 Universal JDBC Driver, collect trace data and run utilities that format the trace data. You should run the trace and diagnostic utilities only under the direction of IBM(R) software support.

Related reference:

- JDBC and SQLJ problem diagnosis with the DB2 Universal JDBC Driver
- Example of tracing under the DB2 Universal JDBC Driver

First failure data capture

First-failure data capture (FFDC) is a general term applied to the set of diagnostic information that DB2(R) captures automatically when errors occur. This information reduces the need to reproduce errors to get diagnostic information.

FFDC information can be found in the following files:

Administration Notification Logs:

When significant events occur, DB2 writes information to the administration notification log. The information is intended for use by database and system administrators. Many notification messages provide additional information to supplement the SQLCODE that is provided. The type of event and the level of detail of the information gathered are determined by the NOTIFYLEVEL configuration parameter.

db2diag.log:

Diagnostic information about errors is recorded in this text log file. This information is used for problem determination and is intended for DB2 customer support. The level of detail of the information is determined by the DIAGLEVEL configuration parameter.

db2dasdiag.log:

Diagnostic information about errors encountered specifically in the DAS (DB2 Administration Server) is recorded in this text file.

Dump files:

For some error conditions, extra information is logged in external binary files named after the failing process ID. These files are intended for use by DB2 customer support.

Trap files:

The database manager generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception.

Core files (UNIX only):

When DB2 terminates abnormally, the operating system generates a core file. The core file is a binary file that contains information similar to the DB2 trap files. Core files may also contain the entire memory image of the terminated process.

Related reference:

- Interpreting the administration and error logs
- Dump files
- Trap files
- Interpreting platform specific error logs
- notifylevel - Notify level configuration parameter
- diaglevel - Diagnostic error capture level configuration parameter

First failure data capture locations

The DIAGPATH variable, specified in the database manager configuration, gives the fully qualified path to the first failure data capture (FFDC) storage directory.

The default value for DIAGPATH is a null string. It is recommended that you keep this default value. If you choose to change the value, it is recommended that you use a centralized location, especially if there are multiple database instances.

To change the DIAGPATH value, use the following command:

```
DB2 UPDATE DBM CFG USING DIAGPATH <path>
```

The default value of NULL means that the first failure data capture (FFDC) information is placed in the following locations:

- For Windows® systems:
  - If the DB2INSTPROF environment variable is not set: *db2path\db2instance* (where *db2path* is the path referenced in the DB2PATH environment variable, and *db2instance* is the environment variable containing the ID of the instance owner).
  - If the DB2INSTPROF environment variable is set: *x:\db2instprof\db2instance*, where *x* is the drive referenced in the DB2PATH environment variable, *db2instprof* is the instance profile directory, and *db2instance* is the environment variable containing the ID of the instance owner.

  Note: On Windows NT®, Windows 2000 and Windows XP systems, the DB2 administration notification log is found in the event log and can be reviewed through the Windows Event Viewer. On other operating systems, the administration notification log for the instance is called instance_name.nfy. The diagnostics log (db2diag.log) is still located in the DIAGPATH path.
- For UNIX® operating systems:
  - `$HOME/sqllib/db2dump`, where `$HOME` is the home directory of the instance owner.

We recommend that you clean out the DIAGPATH directory periodically to keep it from becoming too large.

Related reference:

- diagpath - Diagnostic data directory path configuration parameter

Setting the error capture level for the administration notification log file

The information that DB2 records in the administration notification log is determined by the NOTIFYLEVEL setting. To check the current setting, issue:

```
DB2 GET DBM CFG
```

Look for the following variable:

```
Notify Level                             (NOTIFYLEVEL) = 3
```

The meaning of the different notification levels is defined in notifylevel - Notify level configuration parameter.

To alter the setting, use the command:

```
DB2 UPDATE DBM CFG USING NOTIFYLEVEL X
```

where X is the desired notification level.

Interpreting administration notification log file entries

Use a text editor to view the administration notification log file on the machine where you suspect a problem to have occurred. The most recent events recorded are the furthest down the file. Generally, each entry contains the following parts:

- A timestamp
- The location reporting the error. Application identifiers allow you to match up entries pertaining to an application on the logs of servers and clients.
- A diagnostic message (usually beginning with ″DIA″ or ″ADM″) explaining the error.
- Any available supporting data, such as SQLCA data structures and pointers to the location of any extra dump or trap files.

If the database is behaving normally, this type of information is not important and can be ignored.

- The Administration logs grow continuously . When they get too large, back them up and then erase the file. A new set of files is generated automatically the next time they are required by the system.

The following example shows the header information for a sample log entry, with all the parts of the log identified.

Note: Not every log entry will contain all of these parts.

```
2002-02-05-03.14.39.020766 1    Instance:db2inst1 2   Node:000 3
PID:89198(db2agent (MYDB    )) 4   Appid:*LOCAL.db2inst1.020205091435 5
recovery manager 6  sqlpresr 7  Probe:1 8   Database:MYDB  9
ADM1530E 10  Crash recovery has been initiated. 11
```

Legend:

1. A timestamp for the message.
2. The name of the instance generating the message.
3. For multi–partition systems, the partition generating the message. (In a non-partitioned database, the value is ″000″.)>
4. The DB2 component that is writing the message. For messages written by user applications using the db2AdminMsgWrite API, the component will read “User Application”.

5. Identification of the application for which the process is working. In this example, the process generating the message is working on behalf of an application with the ID *LOCAL.db2inst1.020205091435.

   To identify more about a particular application ID, either:

   - Use the **db2 list applications** command on a DB2 UDB server or **db2 list dcs applications** on a DB2 UDB Connect gateway to view a list of application IDs. From this list, you can determine information about the client experiencing the error, such as its node name and its TCP/IP address.
   - Use the **db2 get snapshot for application** command to view a list of application IDs.
6. The DB2 component that is writing the message. For messages written by user applications using the `db2AdminMsgWrite` API, the component will read “User Application”.
7. The name of the function that is providing the message. This function operates within the DB2 subcomponent that is writing the message. For messages written by user applications using the `db2AdminMsgWrite` API, the function will read “User Function”.

   To find out more about the type of activity performed by a function, look at the fourth letter of its name. In this example, the letter ″p″ in the function ″sqlpresr″ indicates a data protection problem. (Logs could be damaged, for example.)

   The following list shows some of the letters used in the fourth position of the function name, and the type of activity they identify:

b	Buffer pools
c	Communication between clients and servers
d	Data management
e	Engine processes
o	Operating system calls (such as opening and closing files)
p	Data protection (such as locking and logging)
r	Relational database services
s	Sorting
x	Indexing

8. Unique internal identifier. This number allows DB2 customer support and development to locate the point in the DB2 source code that reported the message.
9. The database on which the error occurred.
10. When available, a message indicating the error type and number as a hexadecimal code.
11. When available, message text explaining the logged event.

Example Log Files

Example 1

A power outage causes your DB2 server machine to reboot. While rebooting, some of the file systems do not remount properly.

You want to check on your database’s integrity after the outage, so you start the instance and connect to the database. The connection is successful. In the administration notification log, you see the following entries:

```
2002-02-05-03.14.33.576365   Instance:db2inst1   Node:000
PID:140546(db2star2)   Appid:none
base sys utilities  startdbm Probe:911
ADM7513W  Database manager has started.

2002-02-05-03.14.38.559911   Instance:db2inst1   Node:000
PID:89198(db2agent (MYDB    ))   Appid:*LOCAL.db2inst1.020205091435
buffer pool services  sqlbStartPoolsErrorHandling Probe:39
ADM6080E  The tablespace "TS1" (ID "3"), was put OFFLINE and in
ROLLFORWARD_PENDING. Tablespace state is 0x"00004080".  1

*
2002-02-05-03.14.39.020766   Instance:db2inst1   Node:000
PID:89198(db2agent (MYDB    ))   Appid:*LOCAL.db2inst1.020205091435
recovery manager  sqlpresr Probe:1   Database:MYDB  2
ADM1530E  Crash recovery has been initiated.  3

*
2002-02-05-03.14.44.524546   Instance:db2inst1   Node:000
PID:89198(db2agent (MYDB    ))   Appid:*LOCAL.db2inst1.020205091435
recovery manager  sqlpresr Probe:350   Database:MYDB
ADM1533W  Database has recovered.  However, one or more tablespaces are
offline.  4

*
2002-02-05-03.14.44.956773   Instance:db2inst1   Node:000
PID:89198(db2agent (MYDB    ))   Appid:*LOCAL.db2inst1.020205091435
recovery manager  sqlpresr Probe:370   Database:MYDB
ADM1531E  Crash recovery has completed successfully.  5
```

Legend:

1. Table space ″TS1″ has been put offline and is marked roll-forward pending. This has probably happened because TS1 was located on one of the file systems that did not get remounted during the machine's reboot.
2. The name of the database being recovered was MYDB.
3. A notification that database crash recovery has initialized.
4. Crash recovery indicates that the database was recovered but one or more table spaces were not recovered because they were offline. The table space that was offline is indicated in an earlier message (1).
5. A notification that the database crash recovery was successful.

From here, the **db2 list tablespace containers** command can identify the file systems associated with table space TS1. Your system administrator can put the table space back online by remounting the file systems. You can then roll forward the tablespace.

Example 2

While attempting to create a new table, the following SQLCODE is returned:

```
SQL0968C The file system is full. SQLSTATE=57011
```

The SQLCODE documentation suggests that you examine the administration notification for additional details. Looking in the administration notification log, you see the following message:

```
2002-02-05-13.21.56.270714   Instance:db2inst1   Node:000
PID:88170(db2agent (FOO     ))   Appid:*LOCAL.db2inst1.020205182029
buffer pool services  sqlbSMSDirectWrite Probe:99   Database:FOO
ADM6017E  Tablespace "SYSCATSPACE" (ID "0") is full.  Detected on container
"/home/db2inst1/db2inst1/NODE0000/SQL00001/SQLT0000.0" (ID "0").  Please check
the underlying file system space or user limits.
```

The message indicates the container on which the file system error occurred. Using this information, ensure that there is enough file space on the container. If the file space is sufficient, then ask your system administrator to investigate any system resource limits. For example, on UNIX®, the **ulimit** command can restrict the maximum file size that a user can access.

Related reference:

- Administration logs, error logs and first failure data capture
- Location of logs and files
- notifylevel - Notify level configuration parameter.

Setting the diagnostic log file error capture level

The DB2 diagnostic log is a file that contains text information logged by DB2(R). This information is used for problem determination and is intended for DB2 customer support. See Location of logs and files for details on its location.

The information that DB2 records in the db2diag.log is determined by the DIAGLEVEL setting. To check the current setting, issue:

```
DB2 GET DBM CFG
```

Look for the following variable:

```
Diagnostic error capture level              (DIAGLEVEL) = 3
```

The meaning of the different notification levels is defined in diaglevel - Diagnostic error capture level configuration parameter

To alter the setting, use the command:

```
DB2 UPDATE DBM CFG USING DIAGLEVEL X
```

where X is the desired notification level.

Note: If you are diagnosing a problem that can be reproduced, it is recommended that you use DIAGLEVEL 4 while performing the problem determination.

Interpreting diagnostic log file entries

Use a text editor to view the diagnostic log file on the machine where you suspect a problem to have occurred. The most recent events recorded are the furthest down the file.

Note: The Administration logs grow *continuously*. When they get too large, back them up and then erase the file. A new set of files is generated automatically the next time they are required by the system.

The following example shows the header information for a sample log entry, with all the parts of the log identified.

Note: Not every log entry will contain all of these parts. Only the first several fields (timestamp to TID) and FUNCTION will be present in all the db2diag.log records.

```
2005-01-20-00.28.11.406000-480 1  I2841H435 2       LEVEL: Error 3
PID     : 740 4                TID  : 2640 5    PROC : db2syscs.exe 6
INSTANCE: DB2 7                NODE : 000 8     DB   : SAMPLE 9
APPHDL  : 0-7 10                APPID: *LOCAL.DB2.050120082811 11
FUNCTION: 12 DB2 UDB, data protection, sqlpsize, probe:20
RETCODE : 13 ZRC=0x860F000A=-2045837302=SQLO_FNEX "File not found."
          DIA8411C A file "" could not be found.
```

Legend:

1. A timestamp and timezone for the message.

2. The record ID field. The db2diag.log’s recordID specifies the file offset at which the current message is being logged (e.g. ″2841″) and the message length (e.g. ″435″) for the platform where the DB2 diagnostic log was created.

3. The diagnostic level associated with an error message. e.g. `Info`, `Warning`, `Error`, `Severe`, or `Event`

4. The process ID

5. The thread ID

6. The process name

7. The name of the instance generating the message.

8. For multi-partition systems, the partition generating the message. (In a non-partitioned database, the value is ″000″.)

9. The database name.

10. The application handle. This value aligns with that used in db2pd output and lock dump files. It consists of the coordinator partition number followed by the coordinator index number, separated by a dash.

11. Identification of the application for which the process is working. In this example, the process generating the message is working on behalf of an application with the ID *LOCAL.db2inst1.020205091435.

A TCP/IP-generated application ID is composed of three sections

1. **IP address**: It is represented as a 32-bit number displayed as a maximum of 8 hexadecimal characters.
2. **Port number**: It is represented as 4 hexadecimal characters.
3. A **unique identifier** for the instance of this application.

Note: When the hexadecimal versions of the IP address or port number begin with 0-9, they are changed to G-P respectively. For example, ″0″ is mapped to ″G″, ″1″ is mapped to ″H″, and so on. The IP address, AC10150C.NA04.006D07064947 is interpreted as follows: The IP address remains AC10150C, which translates to 172.16.21.12. The port number is NA04. The first character is ″N″, which maps to ″7″. Therefore, the hexadecimal form of the port number is 7A04, which translates to 31236 in decimal form.

To identify more about a particular application ID, either:

- Use the **db2 list applications** command on a DB2 UDB server or **db2 list dcs applications** on a DB2 UDB Connect gateway to view a list of application IDs. From this list, you can determine information about the client experiencing the error, such as its node name and its TCP/IP address.
- Use the **db2 get snapshot for application** command to view a list of application IDs.
- Use the **db2pd -applications -db sample** command.

12. The product name (″DB2 UDB″), component name (″data protection″), and function name (″sqlpsize ″) that is writing the message (as well as the probe point (″20″) within the function).

13. The return code (if any) returned by a called function. This field consists of a type ("ZRC"), the return code value, and the corresponding error description.

Note: Timestamps in the db2diag.log contain a time zone in DB2 version 8.2. For example: 2005-01-20-00.28.11.406000-480, where "-480" is the difference between UTC (Coordinated Universal Time, formerly known as GMT) and local time at the application server **in minutes.** Thus -480 represents UTC - 8 hours, i.e. PST (Pacific Standard Time).

Now that you have seen a sample db2diag.log entry, here is a list of all of the possible fields:

```
<timestamp><timezone>       <recordID>        LEVEL: <level> (<source>)
PID     : <pid>             TID  : <tid>       PROC : <procName>
INSTANCE: <instance>        NODE : <node>      DB   : <database>
APPHDL  : <appHandle>       APPID: <appID>
FUNCTION: <prodName>, <compName>, <funcName>, probe:<probeNum>
MESSAGE : <messageID>  <msgText>
CALLED  : <prodName>, <compName>, <funcName>   OSERR: <errorName> (<errno>)
RETCODE : <type>=<retCode> <errorDesc>
ARG #N  : <typeTitle>, <typeName>, <size> bytes
... argument ...
DATA #N : <typeTitle>, <typeName>, <size> bytes
... data ...
```

Interpreting the db2diag.log file informational record

The first message in db2diag.log should always be an informational record.

The only exception is the case when the first message in a log file is produced by a component using ossLog API calls (e.g. the DAS, genreg, etc) in which case there will not be an informational record output.

An example of an informational record is as follows:

```
2004-08-08-19.43.54.155770-240 I1A1044            LEVEL: Event
PID     : 5710078              TID  : 1           PROC : gtf
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START   : New db2diag.log file
DATA #1 : Build Level, 144 bytes
Instance "db2inst1" uses "64" bits and DB2 code release "SQL08020"
with level identifier "03010106".
Informational tokens are "DB2 v8.1.1.64", "n040805", "U498350", FixPak "7".
DATA #2 : System Info, 192 bytes
System: AIX titanic_1 2 5 0029317A4C00
CPU: total:32 online:8
Physical Memory(MB): total:32000 free:60
Virtual  Memory(MB): total:71936 free:38311
Swap     Memory(MB): total:39936 free:38251
Kernel   Params: msgMaxMessageSize:4194304  msgMaxQueueSize:4194304
shmMax:68719476736 shmMin:1 shmIDs:131072 shmSegments:0 semIDs:131072
semNumPerID:65535 semOps:1024 semMaxVal:32767 semAdjustOnExit:16384

Information in this record is only valid at the time when this file
was created (refer to this record's time stamp for that information).
```

The Informational record is output for "db2start" on every logical partition. This results in multiple informational records: one per logical partition. Since the informational record contains memory values which are different on every partition, this information may be useful.

Interpreting an SQL structure in the db2diag.log file

For severe errors, an SQLCA structure is dumped into the db2diag.log. For details on the SQLCA fields, see the related topics below.

The following diagram provides an example of a diagnostics log with an SQLCA dump.

```
2003-11-04-12.21.16.559205-300 I381956-688        LEVEL: Severe
PID     : 47918                TID  : 1           PROC : db2agent (SAMPLE)
INSTANCE: payroll              NODE : 000         DB   : SAMPLE
APPHDL  : 0-25                 APPID: *LOCAL.payroll.0E9CB4172114
DATA #1 : SQLCA, PD_DB2_TYPE_SQLCA, 136 bytes    1
sqlcaid : SQLCA    sqlcabc: 136    sqlcode: -980    2    sqlerrml: 0
sqlerrmc:    3
sqlerrp : sqlrita
sqlerrd    4  : (1) 0xFFFFE101     5    (2) 0x00000000    (3) 0x00000000
          (4) 0x00000000    (5) 0x00000000    (6) 0x00000000
sqlwarn : (1)         (2)       (3)       (4)      (5)       (6)
          (7)         (8)       (9)      (10)     (11)
sqlstate:
```

Legend:

1. Beginning of the SQLCA entry.
2. The SQL state (when negative, an error has occurred).
3. Any reason codes associated with the SQL error code.
4. Sometimes there are several errors leading to the final SQL error code. These errors are shown in sequence in the sqlerrd area and may be described by calling `db2diag -rc error_code` if they are ZRC or ECF errors (i.e. start with 0x8 or 0x9).

The hexadecimal representation of an SQL error.

Related reference:

- Administration logs, error logs and first failure data capture
- Location of logs and files
- SQLCA (SQL communications area)
- db2diag - Analyzing db2diag.log files
- diaglevel - Diagnostic error capture level configuration parameter

Dump files

Dump files are created when an error occurs for which there is additional information that would be useful in diagnosing a problem (such as internal control blocks). Every data item written to the dump files has a timestamp associated with it to help with problem determination. Dump files are in binary format and are intended for DB2® customer support representatives.

When a dump file is created or appended, an entry is made in the db2diag.log indicating the time and the type of data written. These db2diag.log entries resemble the following:

```
2004-12-05-06.26.55.664042-360 I5969607C424     LEVEL: Severe
PID     : 646                 TID  : 1          PROC : db2agent (instance) 0
INSTANCE: db2inst1            NODE : 000        DB   : SAMPLE
APPHDL  : 0-9                 APPID: * LOCAL.DB2.050120082811
FUNCTION: DB2 UDB, DRDA Application Server, sqljsSignalHandler, probe:10
MESSAGE : DIA0505I Execution of a component signal handling function has begun.

2004-12-05-06.26.55.696441-360 I5970032C166       LEVEL: Severe
PID:646 TID:1 NODE:000 Title: **** DRDA ASCB ****
Dump File:/home/db2inst1/sqllib/db2dump/6461.000 1
```

Legend:

1 In this UNIX® example, SECTION STMT data is stored in a file named 56772.000 located in the /home/db2/sqllib/db2dump directory.

Notes:

- For partitioned database systems, the file extension identifies the partition number. For example, the following entry indicates that the dump file was created by a DB2 process running on partition 10:

```
Dump File: /home/db2/sqllib/db2dump/56772.010 Data : SECTION STMT
```

Related concepts:

- on page 0
- on page 0
- on page 0
- on page 0
- "System core files (UNIX)" on page 69

Trap files

DB2® generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception.

All signals or exceptions received by DB2 are recorded in the trap file. The trap file also contains the function sequence that was running when the error occurred. This sequence is sometimes referred to as ″function call stack″ or ″stack traceback.″ The trap file also contains additional information about the state of the process when the signal or exception was caught.

DB2 customer support may require trap files for problem analysis. They are located in the directory specified by the DIAGPATH database manager configuration parameter.

On UNIX®-based systems, the first letter in their names is ″t″, followed by a process identifier (PID). The file extension is the partition number (000 on single partition databases).

On Windows® systems, each trap file is named Pxxxxx.yyy where xxxxx is the PID and yyy is the database partition number (or 000 on single partition databases). If the trap file is generated because of an exception, it will have the extension .TRP.

Examples:

- t56772.000 is a trap file for the process with pid 56772.

- t56772.010 is a trap file for the process with pid 56772. It was created by a DB2 process running on partition 10.

Trap file generation:

DB2 can generate a trap files on demand, but you should only do this if requested by DB2 Customer Support.

Refer to **db2nstck and db2_call_stack - Generating EDU call stacks** if you are required to generate trap files.

Related reference:

- db2nstck and db2_call_stack - Generating EDU call stacks
- Administration logs, error logs and first failure data capture
- System core files - UNIX only
- Dr Watson log and Windows event logs

The trap files need not be formatted on UNIX platform.

Formatting trap files (Windows)

A tool called db2xprt.exe is available to let you format trap files (*.TRP). It can be found under the SQLLIB/BIN directory. This tool formats DB2 UDB's binary trap files into a human readable ASCII file. Trap files are located in the instance directory (DB2INSTPROF) by default or in the diagnostic data directory path if the DIAGPATH database manager configuration parameter is set.

Authorization

You must have access to the DIAGPATH directory.

Command syntax

```
 >>-db2xprt--+------------+--+----+--+----+-- infile --+-----------+><
             +-/p-- path -+  '-/m-'  '-/n-'            '- outfile -'
             '-/v-------'
```

Command parameters

/p path
A semicolon (;) separated path that points to the location or locations where the binary files and PDB files are located.

/v Displays version information.

/m Formats a memory dump along with the rest of the trap file.

/n Format data without regard to line number information.

infile Specifies the input file.

outfile
Specifies the output file.

For example, if a trap file called ″DB30882416.TRP″ had been produced in your DIAGPATH, you could format it as follows:

```
db2xprt DB30882416.TRP DB30882416.FMT
```

Analyzing trap files

Before analyzing the stack trace in a trap file, it is important to understand the signal or exception which generated the trap file.

In a trap file on a Windows platform, this can be seen in the trap file as follows:

```
The following information is for pid <3088> tid <2416>
Exception C0000005 Occurred
Exception Address = 6C843B8C
Other Unknown access fault
```

On UNIX platforms, this information will look more like this:

```
Signal #11 (SIGSEGV): si_addr is 0x00000078, si_code is 0x00000033 (SEGV_ACCERR:Invalid permissions

Resource Limits
Data seg top [sbrk(0)] = 0x20736010
Cur data size (bytes)  = 0x000000000EFFFE00
```

These exception and signal numbers can be looked up in operating system header files. For example, ″signal #11 (sigsegv)″ is defined on AIX in /usr/include/sys/signal.h as follows:

```
#define SIGSEGV   11    /* (*) segmentation violation */
```

Common signals and exceptions that cause trap file generation

Below are the most common traps and exceptions for which you will see trap files:

1. Invalid memory accesses. These are indicated by exception C0000005 on Intel platforms, and Signal #11 (SIGSEGV) on Unix. These occur when memory is not mapped (doesn't exist) or when a process does not have permission to perform the desired action (e.g. write to protected memory).
2. Illegal instructions. These are indicated by C000001C on Intel, or Signal #4 (SIGILL) on Unix. They are usually caused by a bad (often NULL) function pointers, stack overwrites (overwriting ″saved″ instruction addresses) or possibly memory corruption.
3. Stack overflow exception. This can occur on Windows platforms, and is marked by an Exception C00000FD. It may also rarely occur on UNIX platforms. It can occur when the stack limit is very low.
4. SIGKILL. This Signal #9 can be caused by someone manually (or through a script) killing a DB2 process, in which case, no trap file is generated.
5. DB2 self-triggered diagnostic trap files (aka ″programming signals″). There are cases where DB2 will force a trap file to be dumped for diagnostic purposes by sending itself a programming signal, but the process does not terminate (the instance is not abnormally ended). The user generated trap files (db2nstck or db2_call_stack) will also have these signal numbers. These traps are indicated by:
   - Signal #36 on AIX
   - Signal #21 on Solaris
   - Signal #29 on HP/UX
   - Signal #23 on LINUX
   - Windows platforms will have a trap file <tid>.000 with an entry stating that no Exception record is present.

   Sometimes these diagnostic trap files are just informative. Other times they are an indication of a more serious condition and will result in an abort or some other ″fatal″ symptom soon after, e.g. a ″real″ trap.

6. SIGABRT. These are triggered by calling abort(). Most SIGABRT (Signal #6) traps on UNIX platforms need not be investigated. There are scenarios where the DB2 instance will intentionally be brought down with an abort() call (Signal #6) if the situation is sufficiently severe. The abort() itself and the resulting trap files and error entries are not a direct symptom of the problem and should not be analyzed. In these cases, DB2 has not really trapped in the true sense. On AIX, errpt -a will indicate a Signal #6, i.e. SIGABRT, which should also produce a core file, i.e. a dump of the process image receiving the signal. In these cases, the root problem may not be a DB2 problem. Certain types of data corruption and other severe errors may cause us to bring ourselves down (aka ″panic″).

Note: You will often see several of these traps files when a SIGSEGV or SIGILL has been encountered on another process. The parent of a trapping process will call abort() to bring the instance down in it's ″SIGCHLD″ signal handler, which essentially means it sends itself a SIGABRT. Thus you will typically see one process performing the illegal operation receiving a SIGSEGV or SIGILL followed by it's parent receiving a SIGABRT.

Analyzing the stack trace back

Once you have determined which trap files are most likely to be pertinent to the problem investigation, it is time to look at the stack traceback (also called ″function sequence″ or ″function call stack″) in the trap file.

Here's an example from an Windows trap file:

```
++++++++++++++++++++++++++++++++++++++++++++
Failing instruction at 6D7969E5 offset: 00000615 in
<?sqler_callbDrdaInput@@YAHPAUUCintfc@@@Z>
<D:/wsdb/db2nt_v81fp3/s030728/engn/include/clientbiconv.h:2067>
++++++++++++++++++++++++++++++++++++++++++++

Stack calling chain:
--EBP -----EIP-----ARGS--
45FAD058 6D7969E5 3D6ECE30 6D790D6A 3D6ECE30 00000000 offset: 00000615 in
<?sqler_callbDrdaInput@@YAHPAUUCintfc@@@Z>
<D:/wsdb/db2nt_v81fp3/s030728/engn/include/clientbiconv.h:2067>

45FAD654 6D790D6A 3D6ECE30 45FAD670 00000000 45FAD6A4 offset: 0000041A in
<?sqlerCallDL@@YAHPAUUCintfc@@PAUUCstpInfo@@@Z>
<sqlerEngineCallbacks.C:1060>

45FAD6A8 6D1A3F5F 3D6ECE30 45FAD790 00001CA3 0000004F offset: 000002E3 in
<?sqljs_ddm_excsqlstt@@YAHPAUUCintfc@@PAUsqljsDDMObject@@@Z>
<sqljsexe.C:1096>

45FAD6CC 6D189182 17108040 45FAD790 3D6ECE30 71C00000 offset: 0000004E in
<?sqljsParseRdbAccessed@@YAHPAUsqljsDrdaAsCb@@PAUsqljsDDMObject@@PAUUCintfc@@@Z>
<sqljspar.C:348>

45FAD7C4 6D189DA8 17108040 3D6ECE30 00000000 171086C6 offset: 00000234 in
<?sqljsParse@@YAHPAUsqljsDrdaAsCb@@PAUUCintfc@@@Z>
<sqljspar.C:761>

45FAD804 6D180AE9 171086C6 000000E7 00AAAC00 00000000 offset: 00001041 in
<?sqljsDrdaAsDriver@@YAHPAUsqlcc_init_struct@@@Z>
<sqljsqam.C:1103>

45FAD838 6D18094C 00000000 00000000 00000000 00000000 offset: 00000EA4 in
<?sqljsDrdaAsDriver@@YAHPAUsqlcc_init_struct@@@Z>
<sqljsqam.C:1597>
```

Without access to DB2 source code, you cannot go much further in your analysis of this information. To determine whether or not you are encountering a known problem, however, you should identify the top two of three functions in the stack traceback and search for existing APARs that address traps on one or all of these functions.

For example in this case, the functions of interest would be:
- ″sqler_callbDrdaInput″
- ″sqlerCallDL″
- ″sqljs_ddm_excsqlstt″

APARs which address problems which had caused DB2 instances to trap should contain information about the stack traceback which typifies the trap. Thus you are able to ascertain whether or not the trap you are encountering is a known problem by searching on those top stack functions. See **How to effectively search for known problems** for more information. In this case, you would find that APAR JR19727 is a very good match for this stack traceback.

If you discover that there are no existing APARs or work-arounds available which match your situation, then at that point you will require assistance from DB2 Customer Support.

Related reference:
- Trap files
- Administration logs, error logs and first failure data capture
- System core files - UNIX only
- Dr Watson log and Windows event logs - Windows only

Platform specific error logs

There are many other files and utilities available outside of DB2 to help analyze problems. Often they are just as important to determining root cause as the DB2 files available. Some of this information is contained in logs and traces within the following areas:
- Operating systems
- Applications and third-party vendors
- Hardware

Based on your operating environment, there may be more places outside of what has been described here, so be aware of all of the potential areas you ma y need to investigate when debugging problems in your system.

Operating systems:

Every operating system has its own set of diagnostic files to keep track of activity and failures. The most common (and usually most useful) is an error report or event log. Here is a list of how this information can be collected:
- AIX: the `/usr/bin/errpt -a` command
- Solaris: `/var/adm/messages*` files or the `/usr/bin/dmesg` command
- Linux: the `/var/log/messages*` files or the `/bin/dmesg` command
- HP-UX: the `/var/adm/syslog/syslog.log` file or the `/usr/bin/dmesg` command
- Windows : the system, security, and application event log files and the `windir\drwtsn32.log` file (where windir is the Windows install directory)

There are always more tracing and debug utilities for each operating system. Refer to your operating system documentation and support material to determine what further information is available.

Applications and third-party vendors:

Each application should have its own logging and diagnostic files. These files will complement the DB2 set of information to provide you with a more accurate picture of potential problem areas.

Hardware:

Hardware devices usually log informatio n into operating system error logs. However, sometimes additional information is required. In those cases, you need to identify what hardware diagnostic files and utilities may be available for piece of hardware in your environment. An example of such a case is when a bad page, or a corruption of some type is reported by DB2. Usually this is reported due to a disk problem, in which case the hardware diagnostics would need to be investigated. Please refer to your hardware documentation and support material to determine what further information is available.

In summary, to completely understand and evaluate a problem, you may need to collect all information available from DB2, your applications, the operating system and underlying hardware. The db2support tool automates the collection of most DB2 and operating system information that you will need, but you should still be aware of any information outside of this that may help the investigation.

Operating system errors

A problem with a system resource may be indicated by a clear error in the db2diag.log or in the Administration Notification log or the db2diag.log. For example:

```
2004-12-14-06.16.09.234346-300 E16317C370        LEVEL: Error (OS)
PID     : 1240500            TID  : 1          PROC : db2test
INSTANCE: db2inst1                NODE : 000
FUNCTION: DB2 UDB, oper system services, sqloxltc_app, probe:15
CALLED  : OS, -, unspecified_system_function
OSERR   : EINVAL (22) "A system call received a parameter that is not valid."
```

In situations where the meaning of the operating system error is not as clearly defined, or when you wish to see more information about the meaning of that error, you must refer to the error definition file on your server.

On most UNIX systems, system errors can be found in /usr/include/sys/errno.h. On Linux, the error numbers are located in /usr/include/asm/errno.h. Here are some of the most common:

```
#define EPERM 1 /* Not super-user */
#define ENOENT 2 /* No such file or directory */
#define EIO 5 /* I/O error */
#define ENOMEM 12 /* Not enough core */
#define EACCES 13 /* Permission denied */
#define ETXTBSY 26 /* Text file busy */
#define EFBIG 27 /* File too large */
#define ENOSPC 28 /* No space left on device */
```

On Windows, system error header files are installed on the system with a compiler or the Windows SDK. Here are a few constants:

```
#define ERROR_FILE_NOT_FOUND 2L
#define ERROR_ACCESS_DENIED 5L
#define ERROR_INVALID_ACCESS 12L
```

You can also invoke the net helpmsg command on Windows, which may provide information about the error., For example,

```
C:\>net helpmsg 5
Access is denied.
```

For both UNIX and Windows platforms, besides searching any problem databases you may have access to, you can use resources on the Web to research errors. Searching the following locations using the error constant as a search keyword (for example "ENOSPC", instead of 28), along with the operating system API being called (if known), will often provide a few hints about the meaning or cause of the error:

- AIX library: http://www-1.ibm.com/servers/aix/library/index.html
- HP-UX: http://docs.hp.com/
- Solaris: http://docs.sun.com/
- Windows: http://support.microsoft.com/
- And, of course, don’t forget the DB2 online support site: http://www.ibm.com/software/data/db2/udb/support

Setting up the system error log (UNIX)

Prerequisites:

You must have root authority.

Procedure:

To route alerts to the system error log, perform the following the steps:

1. Add the following line to the `/etc/syslog.conf` file.

   ```
   user.warn     fully_qualified_file_name
   ```

   where:

   - *user* is the facility to log. This includes DB2 and any other applications logging messages with the same facility name.
   - *warn* is the priority over which messages are logged. Available choices are:

     alert Only alert messages are logged

     err Error and alert messages are logged

     warn Warning, error, and alert messages are logged

     info Information, warning, error, and alert messages are logged
   - *fully_qualified_file_name* is the file (along with its fully qualified path) where the messages will be logged and the SQLCA will be dumped. This file will **not** be created by the system; you must create it in the specified path.
2. Restart the syslog daemon so that it will know about the new configuration in the `syslog.conf` file:
   a. (AIX systems only) Use the **refresh -s syslogd** command.
   b. Use the **kill -1** *pid_of_syslogd* where *pid_of_syslogd* is the process ID of the *syslogd* process. You can obtain this process ID by issuing the **ps -fu syslogd** command.

3. Check to see if information is being logged into the syslog file by issuing:
   ```
   ps -fu db2sysc
   kill -36 db2sysc.process.id
   ```
4. Check the file at *fully_qualified_file_name* (as defined in the /etc/syslog.conf file). If there is information in the file, then the system error log has been enabled to capture the information.

The log file may grow quickly, and you will have to reduce its size periodically. You must use **kill -1** *pid_of_syslog* after you issue the following commands:

```
mv logfile logfile.old
touch logfile
```

On AIX, you can include the following line in the **crontab** that you run as part of your regular system maintenance instead of using the **kill** command:

```
refresh -s syslogd
```

Related concepts:

- on page 0

System core files (UNIX)

If a program terminates abnormally, a core file is created by the system to store a memory image of the terminated process. Errors such as memory address violations, illegal instructions, bus errors, and user-generated quit signals cause core files to be dumped.

The core file is named ″core″, and is placed in the directory where the application was running. Note that system core files are distinct from DB2® core files.

Related concepts:

- "Determining active process status" on page 41

Accessing system core file information (UNIX)

The **dbx** system command helps you determine which function caused a system core file to be created. This is a simple check that will help you identify whether the database manager or DB2 Connect is in error, or whether an operating system or application error is responsible for the problem.

Prerequisites:

You must have the **dbx** command installed.

Procedure:

To determine which function caused the core file dump to occur:

1. Enter the following command from a UNIX-based command prompt:
   ```
   dbx program_name core_filename
   ```

   where *program_name* is the name of the program that terminated abnormally, and *core_filename* is the name of the file containing the core file dump.

   The *core_filename* parameter is optional. If you do not specify it, the default name ″core″ is used.
2. To obtain symbolic information, compile the application using the ″-g″ option.

3. To end the **dbx** command, type **quit** at the dbx prompt.
4. For the HP-UX operating system, use the **xdb** command for similar function.
5. On Version 4.1 of AIX, ensure that the full core option has been enabled using the **chdev** command or smitty.
6. The **dbx** command provides much more function than is described in this section. To find out more, enter `man dbx` from a UNIX-based command prompt.

Example of the dbx Command: The following example shows how to use the **dbx** command to read the core file for a program called ″main″.

1. At a command prompt, enter:

   ```
   dbx main
   ```
2. Output similar to the following appears on your display:

   ```
   dbx version 3.1 for AIX.
   Type 'help' for help.
   reading symbolic information ...
   [using memory image in core]
   segmentation.violation in freeSegments at line 136
   136        (void) shmdt((void *) pcAdress[i]);
   ```
3. The name of the function that caused the core dump is ″freeSegments″. If the function name begins with ″db2″, ″sql″, or ″ddcs″, it may indicate an error in the database manager or DB2 Connect products. Enter **where** at the dbx prompt to display the program path to the point of failure.

   ```
   (dbx) where
   freeSegments(numSegs = 2, iSetId = 0x2ff7f730, pcAddress = 0x2ff7f758, line
   136
   in "main.c"
   main (0x1, 2ff7f7d4), line 96 in "main.c"
   ```

   In this example, the error occurred at line 136 of freeSegments, which was called from line 96 in `main.c`.
4. To end the **dbx** command, type **quit** at the dbx prompt.

Accessing event logs (Windows)

Windows event logs can also provide useful information. While the system event log tends to be the most useful in the case of DB2 crashes or other mysterious errors related to system resources, it is worthwhile obtaining all three types of event logs:

- System
- Application
- Security

The method used to launch the Windows Event Viewer will differ, depending on whether you are using Windows XP, Windows 2003, Windows 2000 or Windows NT.

For example, to open the Event Viewer on Windows XP, click **Start**, click **Control Panel**, click **Performance and Maintenance**, click **Administrative Tools**, and then double-click **Event Viewer**.

Exporting event logs (Windows)

From the event viewer, you can export event logs in two formats

- .evt format, which can be loaded back into an event viewer (for example on another machine) or

- in text format.

Event viewer format is easy to work with since you can use the GUI to switch the chronology order, filter for certain events, and advance forwards or backwards.

Text files provide one significant advantage - you can trust the timestamps! When you export event logs in .evt format, the timestamps are in Coordinated Universal Time and get converted to the local time of the machine in the viewer. If you are not careful, you can miss key events because of time zone differences. Text files are also easier to search, but once you load an event log from another machine into the event viewer, it is easy enough to export it again in text format.

Accessing the Dr. Watson log file (Windows)

The Dr. Watson log, drwtsn32.log, is a chronology of all the exceptions that have occurred on the system. The DB2 trap files are more useful than the Dr. Watson log, though it can be helpful in assessing overall system stability and as a document of the history of DB2 traps. The default path is <install_drive>:\Documents and Settings \All Users\Documents\DrWatson

Combining DB2 and OS Diagnostics

Introduction to system configuration and user environment information:

Diagnosing some problems related to memory, swap files, CPU, disk storage, and other resources requires a thorough understanding of how a given operating system manages these resources. At a minimum, defining resource-related problems requires knowing how much of that resource exists, and what resource limits may exist per user. (The relevant limits are typically for the user ID of the DB2 instance owner.)

Here is some of the important configuration information that you need to obtain:

- Operating system patch level, installed software, and upgrade history
- Number of CPUs
- Amount of RAM
- Swap and file cache settings
- User data and file resource limits and per user process limit
- IPC resource limits (message queues, shared memory segments, semaphores)
- Type of disk storage (for example EMC, Shark, Network Access Storage solution)
- What else is the machine used for? Does DB2 compete for resources?
- Where does authentication occur?

Most platforms have straightforward commands for retrieving resource information. However, you will rarely need to obtain that information manually, since the `db2support` utility collects this data and much more. The detailed_system_info.html file produced by db2support (when the options ″`-s`″ and ″`-m`″ are specified) contains the syntax for many of the operating system commands used to collect this information.

The following exercises are intended to help you discover system configuration and user environment information in various DB2 diagnostic files. The first tutorial familiarizes you with the steps involved in running the db2support utility. Subsequent exercises cover trap files, which provide more DB2-generated data that can be useful in understanding the user environment and resource limits.

Exercise 1: Running the db2support command

1. On each platform that you have available, start DB2 with the db2start command.
2. Assuming you already have the SAMPLE database available, create a directory for storing the output from db2support.
3. Change to that directory and issue:

   ```
   DB2SUPPORT <directory> -D sample -S -M
   ```
4. Review the console output, especially the types of information that are collected.

   You should see output like this (when run on Windows):

   ```
   ...
   Collecting "System files"
        "db2cache.prf"
        "db2dbamr.prf"
        "db2diag.bak"
        "db2eventlog.000"
        "db2imdbd.dmp"
        "db2misc.prf"
        "db2nodes.cfg"
        ...
   Collecting "Detailed operating system and hardware information"
   Collecting "System resource info (disk, CPU, memory)"
   Collecting "Operating system and level"
   Collecting "JDK Level"
   Collecting "DB2 Release Info"
   Collecting "DB2 install path info"
   Collecting "Registry info"
   ...
   Creating final output archive
        "db2support.html"
        "db2_sqllib_directory.txt"
        "detailed_system_info.html"
        "userResponse.xml"
        "db2supp_system.zip"
        "dbm_detailed.supp_cfg"
        "SAMPLE_node0_detailed_db.supp_cfg"
        "dbmsnap.supp_out"
        "db2diag.log"
   db2support is now complete.
    An archive file has been produced: "db2support.zip"
   ```
5. Now use a Web browser to view the detailed_system_info.html file. On each of your systems, identify the following information:
   - Number of CPUs
   - Amount of RAM
   - Operating system level
   - User environment
   - User resource limits (UNIX ulimit command)

Exercise 2: Locating environment information in a DB2 trap file

1. On a UNIX system, ensure DB2 is started, then issue

   ```
   db2_call_stack
   ```

   The call stacks are placed in files in the diagnostic directory (sqllib/db2dump by default). Each EDU has its own file which is called tprocessID.node number.
2. Locate the following in one of the trap files:
   - DB2 code level

- Data seg top (this is the maximum private address space that has been required)
- Cur data size (this is the maximum private address space limit)
- Cur core size (this is the maximum core file limit)
- Signal Handlers (this information may not appear in all trap files)
- Environment variables (this information may not appear in all trap files)
- map output (shows loaded libraries)

Example trap file from AIX (truncated):

```
db2 build information: DB2 v8.1.0.48 s040212 SQL08015
timestamp: 2005-01-21-19.54.15.306950
instance name: db2isnt1.001
EDU name     : db2sysc 1
Signal #36
uname: S:AIX R:2 V:5 M:000786694C00 N:steel
thread id : 1 (0x1)
...
Signal #36: si_pid is 0, si_uid is 0, si_value is 00000000
...
Resource Limits
Data seg top [sbrk(0)] = 0x20116000
Cur data size (bytes)  = 0x000000000EFFFE00
Cur stack size (bytes) = 0x0000000010000000
Cur core size (bytes)  = 0x7FFFFFFFFFFFFFFF
...
```

Correlating DB2 and system events or errors:

System messages and error logs are too often ignored. You can save hours, days, and even weeks on the time it takes to solve a problem if you take the time to perform one simple task at the initial stage of problem definition and investigation. That task is to compare entries in different logs and take note of any that appear to be related both in time and in terms of what resource the entries are referring to.

While not always relevant to problem diagnosis, in many cases the best clue is readily available in the system logs. If we can correlate a reported system problem with DB2 errors we will have often identified what is directly causing the DB2 symptom. Obvious examples are disk errors, network errors, and hardware errors. Not-so obvious are problems reported on different machines, for example domain controllers or NIS servers, which may affect connection time or authentication.

System logs can be investigated in order to assess stability, especially when problems are reported on brand new systems. Intermittent traps occurring in common applications may be a sign that there is an underlying hardware problem.

Here is some other information provided by system logs.

- Significant events such as when the system was rebooted
- Chronology of DB2 traps on the system (and errors/traps/exceptions from other software that is failing)
- Kernel panics, out-of-filesystem-space, and out-of-swap-space errors (which may prevent the system from creating/forking a new process)

System logs can help to rule out crash entries in the db2diag.log as causes for concern. One consistent DB2 crash recovery investigation was resolved by discovering that the system was rebooting - it turned out that the cleaning staff was unplugging the computer every morning at the same time!

7 If you see crash recovery in DB2 Administration Notification or DB2 diagnostic
7 logs with no preceding errors, the DB2 crash recovery is likely a result of a system
7 shutdown.

7 This principle of correlating information extends to logs from any source and to
7 any identifiable user symptoms. For example, it can be very useful to identify and
7 document correlating entries from another application's log even if you can't fully
7 interpret them.

7 The summation of this information is a very complete understanding of your
7 server and of all of the varied events which are occurring at the time of the
7 problem.

7 **Related concepts:**

7 - on page 0

Chapter 2. Specific troubleshooting techniques

Installation issues

This section will help you identify the source of common installation problems.

Note: The first thing you need to do is confirm whether the installation prerequisites are met

Before installing DB2, you should check to make sure that your environment meets the minimum hardware and software requirements as described in the Quick Beginnings documentation. Links to such documentation is included in the Related reference below.

If your system does not meet the minimum requirements, then the DB2 installer could fail. For example, a failure could occur for a simple reason such as not having enough disk space, not having the prerequisite software installed, or kernel parameters not being set according to minimum requirements.

Once you have eliminated the environment or non-DB2 factors, you can focus on the techniques described in subsequent sections.

Related reference:

- Disk and memory requirements for non-partitioned DB2 UDB Enterprise Server Edition (Windows and UNIX)
- Disk and memory requirements for partitioned DB2 UDB Enterprise Server Edition (Windows and UNIX)
- Disk and memory requirements for DB2 UDB clients (Windows and UNIX)
- Installation requirements for non-partitioned DB2 servers (AIX)
- Installation requirements for partitioned DB2 servers (Windows)
- DB2 Information Integrator installation worksheet
- Installation requirements for Query Patroller server (UNIX)
- DB2 PDF and printed documentation

Location of installation error logs

The type of installation error logs produced will depend on the method of installation used.

db2setup wizard:

This DB2 installation program captures all installation information, including errors, in a number of files:

- db2setup.log or db2wi.log file captures the current (or most recent) DB2 installation information including errors.
- db2setup.his or db2.log contains a history of all DB2 installations on your machine
- db2setup.err file captures any error output that is returned by Java (for example, exceptions and trap information)

In some cases you can specify where these files are created, but by default they are located in the following location:

- On Windows operating systems: the "My Documents"\DB2LOG\ directory (the location of the "My Documents" directory will depend on the settings on your computer).
- On UNIX systems: the /tmp directory

db2_install script:

This UNIX installation method generates an error log file in /tmp called db2_install_log.<pid>, where <pid> is the process ID which performed the installation.

Response file installation:

If you are installing via a response file on UNIX, you will be using the "db2setup" command, and the same installation error logs will be produced as described above for the db2setup wizard.

If you are installing via a response file on Windows, you can specify the name and location of the error log using the "/L" flag on the "setup.exe" command. If you do not specify the log file's name, DB2 names it db2.log. The db2.log file is located in the My Documents\db2log folder.

Native installation tools:

Installation error logs may be produced, but the type and location of the file will depend on the tool used. For example, if smit is used, the error log file can usually be found in /smit.log.

Related reference:

- Installation methods for DB2 UDB (Windows and UNIX)
- Administration logs, error logs and first failure data capture
- Interpreting installation error logs

Installation methods for DB2 UDB (Windows and UNIX)

This topic provides information about DB2® UDB installation methods. The following table shows the installation methods that are available by operating system.

Table 1. Installation method by operating system.

Installation method	Windows®	UNIX®
DB2 Setup wizard	Yes	Yes
db2_install script	No	Yes
Response file installation	Yes	Yes
Native installation tools	No	Yes

The following list describes installation methods for DB2 UDB.

DB2 Setup wizard
The DB2 Setup wizard is a GUI installer available on both UNIX and Windows operating systems. The DB2 Setup wizard provides an

easy-to-use interface for installing DB2 UDB and for performing initial setup and configuration tasks. The DB2 Setup wizard can also be used to create instances and response files.

On UNIX systems, the DB2 Setup wizard replaces the text-based installer interface (db2setup).

db2_install script

The db2_install script uses the operating systems's native installation utility to install DB2 UDB. The db2_install script prompts for a DB2 product keyword. This script installs *all* components for the DB2 product you specify, in English only. You cannot select or deselect components or interface language support. The db2_install script does not perform user and group creation, instance creation, or configuration. It installs the DB2 components to your system. This method of installation might be preferred in cases where greater control over the installation setup process than the GUI installer provides is required.

Response file installation

A response file is an ASCII file that contains setup and configuration values. The file is passed to the DB2 setup program and the installation is performed according to the values that have been specified. There are a number of ways to create a response file:

- Using the response file generator (Windows)
- Using the DB2 Setup wizard (UNIX and Windows)
- By customizing sample response files that are provided for each DB2 product (UNIX and Windows)

Using the response file generator, you can create a response file that replicates an existing installation. For example, you might install a DB2 client, fully configure the client, then generate a response file to replicate the installation and configuration of the client to other computers. The response file generator is available only on Windows.

The DB2 Setup wizard can create a response file for both UNIX and Windows installations. The selections you make as you proceed through the DB2 Setup wizard are recorded in a response file that you can save to a location on your system.

For your convenience, you can create a response file without performing an installation. This feature can be useful in an environment where a DBA does not have the authority required to perform an installation. The DBA can create a response file for the installation and provide it to the system administrator who installs the product on the DBA's behalf.

An alternative to using the response file generator or the DB2 Setup wizard to create a response file is to manually modify a sample response file. Sample response files are provided on the DB2 product CD-ROM.

Native installation tools

Installing DB2 using your operating system's native installation provides the greatest control over the installation process, but it is also more difficult than the other installation methods. When installing a particular DB2 product, you have to ensure that the required components are installed and that component dependencies are maintained. Advanced knowledge of both DB2 and your operating environment is required. User and group creation, instance creation, and configuration must be performed manually.

Interpreting Windows installation error logs

db2.log and db2.wi log errors:

Sometimes referred to as an "installation history log file", db2.log contains a history of all DB2 installations that have occurred.

Note: Your installation history log file may not be named "`db2.log`" if you specified a different file name and location using the `-l` parameter on the `setup.exe` command.

The db2wi.log file contains the log of the current (or most recent) installation. Information is written to it as the installation events occur, unlike the history log file where the log is not actually written until the end of the installation. At the end of the installation, the only difference between what is written to the db2wi.log and what is appended to the history log file is that the former also contains a list of Windows Installer properties with current values. This list of properties is very long and is typically only looked at by DB2 Support.

Depending on the problem you are trying to solve, all of the information in the log files may be useful, despite the complexity and amount of information contained in the files. The difficult part is narrowing down the problem and finding the parts of the log files that are most relevant to the problem.

Note: When you run the installation with tracing turned on, the logging is done in verbose mode, thereby creating a log with much more information. Do not confuse the trace file with the log file. They are different files. However, by turning trace on, you increase the logging level of the installation. If you are familiar with Windows Installer you will recognize that the log created when enabling trace is the verbose log that you would typically get by adding the `/l*v` parameter to windows installer based installations.

Ensuring that the log you are reading is the correct log:

In many cases you may find yourself debugging an installation that you did not initiate. If you are unsure whether the log you are viewing is the correct log, there are some hints that you can use to verify that it is the correct log. The best way to determine this is to look at the time of the installation, as well as information about the product being installed. The product shown can be the product code or the path to the installation database used to install the product. If it is the path to the installation database, notice that the file name contains the name of the product. However, if it lists a globally unique identifier (GUID) for example, {D8F53726-C7AD-11D4-9155-00203586D551}, then it is a bit more difficult. Here is a list of the product codes for the DB2 UDB products that you can use to cross-reference information found in the product field.

Product	**ID code**	
DB2 UDB Enterprise Server Edition	{D8F53726-C7AD-11D4-9155-00203586D551}	
DB2 UDB Workgroup Server Edition	{7A28F948-4945-4BD1-ACC2-ADC081C24830}	
DB2 UDB Personal Edition	{C0AA883A-72AE-495F-9601-49F2EB154E93}	

Product	ID code	
DB2 Warehouse Manager	{84AF5B86-19F9-4396-8D99-11CD91E81724}	
DB2 Datalinks Manager	{1D16CA65-F7D9-47E5-BB26-C623A44832A3}	
DB2 Information Integrator Relational Wrappers	{273F8AB8-C84B-4EE6-85E7-D7C5270A6D08}	
DB2 Connect Enterprise Edition	{9C8DFB63-66DE-4299-AC6B-37D799A728A2}	
DB2 Connect Personal Edition	{F1912044-6E08-431E-9B6D-90ED10C0B739}	
DB2 Administration Client	{ABD23811-AA8F-416B-9EF6-E54D62F21A49}	
DB2 Application Development Client	{68A40485-7F7F-4A91-9AB6-D67836E15CF2}	
DB2 Run-time Client	{63F6DCD6-0D5C-4A07-B27C-3AE3E809D6E0}	
DB2 Spatial Extender	{F6846BF9-F4B5-4BB2-946D-3926795D5749}	
DB2 Information Integrator Non-Relational Wrappers	{DD30AEB3-4323-40D7-AB39-735A0523DEF3}	
DB2 Warehouse Manager Connectors	{5FEA5040-22E2-4760-A88C-73DE82BE4B6E}	
DB2 Query Patroller	{7A8BE511-8DF3-4F22-B61A-AF0B8755E354}	
DB2 Cube Views	{C8FEDF8F-84E8-442F-A084-0A0F6A772B52}	
DB2 UDB Express Edition	{58169F10-CA30-4F40-8C6D-C6DA8CE47E16}	
DB2 Run-time Client Lite	{07C9CEE7-4529-4E60-95D3-6B6EF6AC0E81}	

For example, consider the following excerpt of a log file. The date on which the installation took place is February 2, 2005. The GUID is {D8F53726-C7AD-11D4-9155-00203586D551} which matches the DB2 UDB Enterprise Server Edition:

```
=== Verbose logging started: 2/14/2005  15:40:03 Build type: SHIP UNICODE 2.00.2600.1183
Calling process: C:\WINNT\system32\msiexec.exe ===
MSI (c) (8C:88): Resetting cached policy values
MSI (c) (8C:88): Machine policy value 'Debug' is 0
 MSI (c) (8C:88): ******* RunEngine:
******* Product: {D8F53726-C7AD-11D4-9155-00203586D551}
******* Action:
******* CommandLine: **********
```

Determining installation success or failure:

In some cases all you may be interested in is whether the installation was successful. To determine the success or failure of an installation look at the end of the db2wi.log file for a line that looks like the following:

```
MSI (s) (98:8C): Product: DB2 Enterprise Server Edition
 -- Installation operation completed successfully.
```

Likewise, if the installation failed, look for a line at the end of the db2wi.log file that looks like the following:

```
MSI (s) (40:10): Product: DB2 Enterprise Server Edition -- Installation operation failed.
```

Diagnosing major errors:

If you received a failure the next step is to determine the cause of the failure. A general tip that allows you to find the error quickly is to search for Return value 3 in the log file. Once you find this in the log file you will usually see further text detailing what the problem is.

For example, if you had logged into the Windows server using a user ID that does not belong to the Administrators group, and then tried to install DB2, the db2wi.log file would contain an error as follows:

```
...
MSI (s) (40:10): Doing action: VerifyPrereqsCA
Action start 15:47:17: VerifyPrereqsCA.
MSI (s) (40:10): Creating MSIHANDLE (20) of type 790542 for thread 572
1: In order to install the product "DB2 Enterprise Server Edition", the user running the
installation must have at minimum the authority of a member of the user group "Administrators".
Action ended 15:47:59: VerifyPrereqsCA. Return value 3.
MSI (c) (BC:3C): Doing action: SetupCompleteError
Action start 15:47:59: SetupCompleteError.
...
```

In some cases the error comes directly from Windows Installer. In these cases the error can be difficult to understand, but there are methods to get more information about the error. In some cases you may only be presented with an error number along with some strings separated by commas. You can look up these error types directly from the Microsoft Web site at http://www.msdn.microsoft.com

Diagnosing minor errors:

In some cases the installation may complete successfully but with the occurrence of a minor configuration error. When these types of errors occur it means that the installation completed, however an error occurred during the configuration stage of the installation. When this occurs the installation exit code will be set to 1. The most common place to look for these types of errors is during the execution of the action that performs most of the up and running operations. The output from this task can be found by searching for a line that looks like the following:

when installation traces are enabled:

```
Action 10:27:03: DeferredCallURE_CA.
MSI (s) (9C:DC): Executing op:
CustomActionSchedule(Action=DeferredCallURE_CA,ActionType=3137,Source=BinaryData,Target=CallURECA,)
```

```
when installation traces are not enabled:
Action 10:27:03: DeferredCallURE_CA.
```

The up and running portion of the installation which runs during the `Custom Action DeferredCallURE_CA` is organized into tasks. The success status of each is reported to the log which can be used to determine if the task was successful, or if a problem occurred. A successful task will output a line in the log file that looks like the following:

1: Creating/migrating DB2 instances:.......Success

Alternatively, if the task failed you would see a line that looks like the following:

1: Creating/migrating DB2 instances:.......Failure

If you need to find more information about a particular task that failed, or for further details about what the task did, you can look at the lines immediately before the overall result of the task. The following example shows that an instance ″DB2″ was created successfully, and that some DBM Config variables were set. For example:

1: The instance ″DB2″ has been created successfully.

1: The value ″`SVCENAME=db2c_DB2`″ was set in the DBM CFG file for the ″DB2″ instance.

1: The value ″`DB2COMM=TCPIP`″ was set in the Profile Registry for the ″DB2″ instance.

1: Creating/migrating DB2 instances:.......Success

Response file errors:

If you have installed DB2 using a response file, there are some common problems which you may encounter. The most common response file problem is that the installation cannot find the response file that was specified by the `-u` option of setup.exe because the location was specified incorrectly. In a case like this you would see lines in the log file that look like the following:

```
Action start 0:23:55: DetectAndSetInstallPathCA.
Action ended 0:23:55: DetectAndSetInstallPathCA. Return value 1.
Action start 0:23:55: InitSilentInstallCA.
1: Failed to access the response file: "c:\db2ese.rsp".
Action ended 0:23:55: InitSilentInstallCA. Return value 3.
Action ended 0:23:55: INSTALL.
Return value 3.
```

Right before the `Return Value 3`, which shows that the action `InitSilentInstallCA` failed, you can see some information about the error. In this case the response file `c:\db2ese.rsp` cannot be accessed because the path does not exist. This problem can be corrected by simply correcting the response file path given in the `-u` parameter to setup.exe. Another possible cause of response file installation failure can be that the user running the installation does not have permission to access the file.

Keyword errors:

One type of error that can occur is caused by an invalid keyword in the response file installation. The response file that is passed into the installation is validated for two main types of problems before the installation begins. The first type of validation that occurs is for syntax. Certain keywords have length limits, or accept only certain values. If an invalid value is specified for a keyword, or if a keyword is entered that is not recognized, the installation will exit. The second type of validation is semantic. At this stage values are checked to make sure they are compatible with the system and with each other. This stage is only performed if the syntax checking does not find any errors.

Keyword syntax validation:

Response file validation is done during the `InitSilentInstallCA` custom action. To find this you can either search for this method directly (if you know that the problem was with a keyword), or you can use the method described above by searching for Return Value 3. The following examples illustrate some of the error types that might be displayed as a result of the response file syntax validation:

```
MSI (s) (40:54): Doing action: InitSilentInstallCA
Action start 0:27:58: InitSilentInstallCA.
MSI (s) (40:54): Creating MSIHANDLE (5120) of type 790542 for thread 1876
 1: ERROR:A Response file error occurred.
 The value "INVALID_VALUE" is not valid for the keyword "KILL_PROCESSES".

 1: ERROR:A Response file error occurred. Unknown keyword "THIS_IS_AN_INVALID_KEYWORD" at line "116"
 1: ERROR:One or more errors occurred while checking the syntax of the response file "c:\db2ese.rsp"
 Please correct the errors and run the install again.
 1: ERROR:Unable to set the response file "c:\db2ese.rsp" in the up and running engine.
 1: Failed to initialize silent install. Action ended 0:28:05: InitSilentInstallCA.
 Return value 3.
```

In the above example, the keyword for `KILL_PROCESSES` is given a value of `INVALID_VALUE`; however, the only values accepted are `YES` or `NO`. The keyword `THIS_IS_AN_INVALID_KEYWORD` was also in the response file, however this is not recognized by the DB2 Setup wizard.

Response file semantic validation:

Semantic response file validation is also done during the `InitSilentInstallCA` custom action. The errors that occur during the semantic validation are displayed in the log file. After reading the errors that are in the log file it should be apparent what needs to be corrected. For example:

```
Action start 1:51:02: InitSilentInstallCA.
1: ERROR:The TCP/IP entries are invalid.
Please specify a valid service name and port number.
1: ERROR:The response file specified "c:\db2ese.rsp" is not valid.
1: Failed to initialize silent install.
Action ended 1:51:03: InitSilentInstallCA. Return value 3.
```

In this example, the log file indicates that the TCP/IP entries specified in the response file are not valid. The combination of service name and port number conflicted with values that were already on the system.

Related reference:

- Location of installation error logs
- Installation methods for DB2 UDB (Windows and UNIX)
- Tracing installation problems

Interpreting UNIX installation error logs

db2setup.log, db2setup.his and db2setup.err:

The db2setup.log file contains the log of the current (or most recent) installation. The information from the db2setup.log is appended to the db2setup.his (sometimes referred to as an ″installation history log file″) once the installation has ended.

The db2setup.err file captures any error output that is returned by Java (for example, exceptions and trap information).

Depending on the problem you are trying to solve, all of the information in the log files may be useful, despite the complexity and amount of information contained in the files. The difficult part is narrowing down the problem and finding the parts of the log files that are most relevant to the problem.

An example of what will appear in a db2setup.log is as follows:

```
DB2 Setup log file started at: Fri Feb 18 15:42:22 EST 2005 EST
=============================================================

Operating system information: AIX 5.2
Product to install:                 DB2 Universal Database Enterprise Server Edition
Installation type:                  Typical
Previously Installed Components:
    Java(TM) Runtime Environment (JRE)
    Java(TM) Development Kit (JDK)
Selected Components:
    Product Messages - en_US.iso88591
    Base Client Support
    DB2 LDAP Support
...
Enabling Asynchronous I/O:.......Success
Checking license agreement acceptance:.......Success
Command to be run:
"/usr/sbin/installp -acgqX  -d
'/net2/d_v8_inst30/v8_inst30/db2aix5L64_v82/.s040812/ese.sbcsaix1/db2/aix' 'db2_08_01.msg.en_US.is
...
```

The appearance of the subsequent section of the db2setup.log will depend on the platform, since DB2 will use the operating system's native installation utility. Once all of the filesets have been processed, the following messages in the db2setup.log will indicate whether the installation has completed successfully:

```
Installing DB2 file sets:.......Success
Registering DB2 licenses:.......Success
Setting default global profile registry variables:.......Success
Creating the DB2 Administration Server:.......Success
Initializing instance list:.......Success
Creating DB2 instances:.......Success
Updating existing DB2 instances:.......Success
Configuring the DB2 Administration Server:.......Success
Updating global profile registry:.......Success

DB2 Setup log file finished at: Fri Feb 18 16:36:52 2005 EST
=============================================================
```

db2_install_log.<pid>:

The `db2_install_log.<pid>` will be produced when you perform the installation using the `db2_install` script. The log will contain a list of all of the filesets processed and whether or not they were applied successfully. The format of the file will vary, since `db2_install` uses the operating system's native installation utility. For example, on AIX the entries for each fileset would appear as follows:

```
      Pre-installation Verification...
+-----------------------------------------------------------------------+
Verifying selections...done
Verifying requisites...done
Results...

SUCCESSES
---------
 Filesets listed in this section passed pre-installation
 verification and will be installed.

  Selected Filesets
  -----------------
  db2_08_01.db2.rte 8.1.1.64                 # Run-time Environment
  << End of Success Section >>

...

+-----------------------------------------------------------------------+
    Installing Software...
+-----------------------------------------------------------------------+

installp: APPLYING software for:
 db2_08_01.db2.rte 8.1.1.64

. . .

Finished processing all filesets.  (Total time:  19 secs).

+-----------------------------------------------------------------------+
    Summaries:
+-----------------------------------------------------------------------+

Installation Summary
--------------------
Name                        Level           Part        Event       Result
-------------------------------------------------------------------------
db2_08_01.db2.rte           8.1.1.64        USR         APPLY       SUCCESS
+-----------------------------------------------------------------------+
```

To determine the success or failure of the installation, look at the end of the file for a line that looks like the following:

```
The installation logfile can be found in /tmp/db2_install_log.91190.
db2_install program completed successfully.
```

Response file errors:

If you have installed DB2 using a response file, there are some common problems which you may encounter. The most common response file problem is that the installation cannot find the response file that was specified by the `-r` option of db2setup because the location was specified incorrectly. In a case like this, the installation would fail and you would see lines in the db2setup.log file that look like the following:

```
DB2 Setup log file started at: Fri Feb 18 17:36:56 2005 EST
============================================================

Operating system information: AIX 5.2
ERROR:DB2 Setup is unable to open the response file "/home/db2inst1/db2ese.rsp".


DB2 Setup log file finished at: Fri Feb 18 17:36:57 2005 EST
============================================================
```

In this case the response file `c:\db2ese.rsp` cannot be accessed because an incorrect response file path was specified in the db2setup command. Simply verifying the correct path and re-running the db2setup command with the correct information can correct this problem. Another possible cause of response file installation failure can be that the user running the installation does not have permission to access the file.

Keyword errors:

One type of error that can occur is caused by an invalid keyword in the response file installation. The response file that is passed into the installation is validated for two main types of problems before the installation begins. The first type of validation that occurs is for syntax. Certain keywords have length limits, or accept only certain values. If an invalid value is specified for a keyword, or if a keyword is entered that is not recognized, the installation will exit. The second type of validation is semantic. At this stage values are checked to make sure they are compatible with the system and with each other. This stage is only performed if the syntax checking does not find any errors.

Keyword syntax validation:

The following example illustrates one of the error types that might be displayed is an error is encountered during the response file syntax validation:

```
The response file specified "/home/lcawley/db2ese.rsp" is not valid.
For more information please see the DB2 installation log at  "/tmp/db2setup.log".
```

The db2setup.log shows the following:

```
DB2 Setup log file started at: Fri Feb 18 18:56:15 2005 EST
============================================================

Operating system information: AIX 5.2
ERROR:A Response file error occurred. The value "MAYBE" is not valid for the
keyword "LIC_AGREEMENT".

ERROR:One or more errors occurred while checking the syntax of the response
file "/home/lcawley/db2ese.rsp". Please correct the errors and run the install
again.


DB2 Setup log file finished at: Fri Feb 18 18:56:18 2005 EST
============================================================
```

In the above example the keyword for LIC_AGREEMENT is given a value of MAYBE, however the only values accepted are ACCEPT or DECLINE. This problem can be resolved by correcting the value in the response file and running db2setup again.

Response file semantic validation:

The following example illustrates one of the error types that might be displayed is an error is encountered during the response file syntax validation:

```
DBI1702E The specified service name or port number conflicts
with existing values in the TCP/IP services file.
...
The response file specified "/home/lcawley/db2ese.rsp" is not valid.
For more information please see the DB2 installation log at "/tmp/db2setup.log".
```

The db2setup.log shows the following:

```
DB2 Setup log file started at: Fri Feb 18 19:06:14 2005 EST
============================================================

Operating system information: AIX 5.2
ERROR:DBI1702E The specified service name or port number conflicts
          with existing values in the TCP/IP services file.

Explanation:

The service name or port number entered by the user conflicts with
existing values in the TCP/IP services file. The service name may
already be used with a different port number, or the port number
may already be used with a different service name.
```

```
User Response:

Specify a service name and port number that does not conflict
with existing entries in the services file.

DB2 Setup log file finished at: Fri Feb 18 19:06:19 2005 EST
============================================================
```

After reading the errors that are in the log file it should be apparent what needs to be corrected. In this example, the log file indicates that the TCP/IP entries specified in the response file are not valid. The combination of service name and port number conflicted with values that were already defined on the system.

Related reference:

- Location of installation error logs
- Installation methods for DB2 UDB (Windows and UNIX)
- Tracing installation problems

Tracing installation problems

- setup - Install DB2 Command
- db2setup - Install DB2 Command

Troubleshooting a DB2 Information Integrator installation

Instance creation and update issues

Determining the cause of instance creation problems

If the problem with instance creation occurs during DB2 installation, then you need to review the error message and the installation error logs . Forexample, if you neglect to drop the instances prior to uninstalling DB2 on UNIX-based systems, `db2setup` may return the following error when you attempt to reinstall DB2 and recreate the instance:

```
The user you selected already has an sqllib directory in their home directory.  This user cannot be
```

In this situation, you need to verify whether the sqllib directory (e.g. `/home/db2inst1/sqllib` does indeed exist. It is possible in this situation that removing the sqllib directory will not be sufficient to resolve the problem, since if you did not drop the instances prior to uninstalling DB2, they may still appear in the DB2 UDB Instance Profile Registry. Such a situation would result in errors in the installation error log and a failure to create the instance. An example of such errors from db2setup.log on AIX is as follows:

```
Installing DB2 file sets:.......Success
Registering DB2 licenses:.......Success
Setting default global profile registry variables:.......Success
Creating the DB2 Administration Server:.......Success
ERROR:Could not switch current DB2INSTANCE to "db2inst1". The return code is
"-2029059916".

Initializing instance list:.......Success
ERROR:DBI1122E Instance db2inst1 cannot be updated.
```

Explanation:

An attempt was made to update an instance. This instance cannot be updated because:

- This "db2iupdt" command cannot be used to update this instance.
- The instance is still active.

User Response:

Ensure that you are using the correct version of the "db2iupdt" command. Also ensure that there are no db2 processes running at the instance. Retry the command.

DBI1079I Output is saved in the log file /tmp/db2iupdt.log.113908.

In this case you may need to manually remove the old instance from the instance profile registry and then re-attempt the installation. In this example, that would mean removing "db2inst1" from the list of instances in the /var/db2/V8.1/profiles.reg file. This problem can be avoided of course if you ensure that you follow the instructions when uninstalling DB2 and drop the instances prior to uninstalling the DB2 product.

If your instance creation problem is not occurring during installation, or if everything other than the instance creation succeeded during the installation, you should try to debug the problem using either the db2isetup tool (on UNIX platforms only) or the db2icrt script.

Additional trace information can be obtained when using db2icrt and db2isetup on UNIX-based systems.

Example 1:

```
DB2DIR/instance/db2icrt -d -u db2fenc1 db2inst1
```

where DB2DIR is the DB2 installation directory, will produce additional trace points in the /tmp/db2icrt.log file.

Example 2:

```
DB2DIR/instance/db2isetup -t /home/db2inst1/isetup.trc
```

where DB2DIR is the DB2 installation directory, will produce a trace file as well as a default /tmp/db2icrt.log file.

Related reference:

- Location of installation error logs
- Creating an instance using db2icrt
- db2isetup - Start Instance Creation Interface Command
- Setting environment variables on UNIX systems

Identifying instance update issues (UNIX)

Since most instance creation and update problems happen in the UNIX environment, this section focuses only on the UNIX platform. Note however that the command does exist in DB2 Enterprise Server Edition environments on Windows.

To identify an instance update issue, you should first check the diagnostic log file /tmp/db2iupdt.log.pid. For example, if you attempt to update an instance called db2inst1 as follows:

```
db2iupdt -u db2fenc1 db2inst1
```

But it fails with ″DBI1069E Unexpected error. Function = terminate_instance, Return code = 13.″, you should check the /tmp/db2iupdt.log.<pid>. The error log′s full name will be indicated at the time of the error. For example:

```
DBI1079I Output is saved in the log file /tmp/db2iupdt.log.49998.
```

Explanation:

All processed and failed operations have been saved into this log file.

User Response:

Do not modify this file in any way. This file is for IBM Technical Support reference.

Here is an example of the db2iupdt.log:

```
     Program name  = db2istop
     Instance name = db2inst1
     Home dir      = /home/db2inst1
     Version       = 81
     Product dir   = /usr/opt/db2_08_01
     db2stop return code = SQL6033W
     exit code = 13
DBI1069E Unexpected error. Function = terminate_instance, Return code =  13.
```

Explanation:

An unexpected error occurred during the execution of this program.

...

```
DBI1250E Applications are still using instance db2inst1.
```

Explanation:

There are applications that are still running that are using the specified instance. All applications using this instance must be terminated before the command can be completed successfully. You can get a list of the applications that are currently using the instance by issuing the command:

```
    db2 list applications
```

User Response:

You can either wait for the applications to end by themselves, or you can explicitly force the applications to end. You can logon as the instance owner and run the command

...

```
DBI1122E Instance db2inst1 cannot be updated.
```

...

```
DBI1079I Output is saved in the log file /tmp/db2iupdt.log.49998.
```

Explanation:

All processed and failed operations have been saved into this log file.

Based on the above error log, the causeof the instance update problem is that there are applications still using the instance at the time that the "`db2iupdt`" was attempted.The actions `db2iupdt.log is very clear in this case.`

If the actions necessary are not this clear, you can debug this further by running `db2iupdt` with the `-d` debug option. In order to redirect the debug information to a file, you may use shell redirection.

For example:

```
db2iupdt -d  db2inst1 2>&1 | tee db2iupdt.debug.out
```

The output file in this example is called `db2iupdt.debug.out`. Here is an example of the output you would see in `db2iupdt.debug.out` file for the same error as was described previously:

```
...
+ [ 0 -eq 0 ]
+ /usr/bin/echo ## exit function terminate_instance
+ /usr/bin/tee -a /tmp/db2iupdt.log.47370
## exit function terminate_instance
+ return 13
status=13
+ [ 13 -ne 0 ]
...
+ /usr/bin/ps -elf
+ /usr/bin/grep db2inst1
200001 A   root  4948 47370  1  60 20 1b83   228 30e5cdd0 20:46:15  pts/2  0:00 grep db2inst1
 40001 A  db2inst1  9820 41098  0  60 20 b855   552 31e311cc 20:46:06   -  0:00 db2loggw (LISE) 0
 40001 A  db2inst1 18806 41098  0  60 20 c959   648 c0041500 20:46:08   -  0:00 db2pfchr 0
 40001 A  db2inst1 26426 41098  0  60 20 2de7   608 31e31a0c 20:46:05   -  0:00 db2loggr (LISE) 0
 40001 A  db2inst1 31840 54756  0  60 20  5a3  1144          20:45:20   -  0:00 db2tcpcm 0
...
 40001 A  db2inst1 54756 30608  0  60 20 89f0  1416          20:45:19   -  0:00 db2sysc 0

+ /usr/bin/ipcs -a
+ /usr/bin/grep db2inst1
q  8126477 0xffffffff --rw-------  db2inst1    usr  db2inst1    usr       0       0
4194304      0      0 no-entry no-
...
+ display_msg /usr/opt/db2_08_01/msg/en_US.iso88591/db2install.cat 250
DBI1250E Applications are still using instance %s.\n db2inst1
+
```

From the above output, the `terminate_instance` function returnsa value of 13. If you look at the scripts in the `DB2DIR/instance` directory, where `DB2DIR` represents `/usr/opt/db2_08_01` on AIX, and `/opt/IBM/db2/V8.1` on all other UNIX-based systems, you will see that `terminate_instance` is defined in the `db2iutil` script.

Note: If you have a FixPak or modification level installed in an alternate path, the `DB2DIR` directory is `usr/opt/db2_08_FPn` on AIX and `opt/IBM/db2/V8.FPn` on all other UNIX-based systems, where n represents the number of the FixPak or modification level.

Within `db2iutil` you can see that `terminate_instance` stops all applications, CLP backend processes and the database manager for a specified instance.

The return code (13) is obtained from the `db2istop` command here:

```
# Stop all applications and the database manager for the instance.
status=${FALSE?}
/usr/bin/su ${sysadm?} -c "${PROGDIR?}/db2istop ${argforce?} \
       ${nodenum?} ${instname?} ${insthome?} ${db2version?} \
       ${db2proddir?} ${TMPFILE3?}"
status=$?
```

The `db2istop` script shows the meaning of the value 13:

```
EXIT_DB2STOP_ERR=13
```

Other `db2iupdt` errors on UNIX can be investigated using the log file and debug traces in the same manner.

Related reference:

- Updating instance configuration on UNIX
- Updating instance configuration on Windows

Avoiding fixpak upgrade problems

Since most FixPak upgrade problems can be avoided by simply following the steps as outlined in the FixPak readme, you should review it prior to doing the FixPak upgrade.

As a general rule, you need to do the following things before installing a FixPak:

- Verify that there is enough disk space on your system.
- Verify that the required prerequisites are installed (as listed in the FixPak readme).
- Stop all database processes (i.e. stop all instances as well as the DB2 Administration Server, the Fault Monitor and the License Daemon).
- For AIX only, unload the DB2 libraries with `/usr/sbin/slibclean` as root.

If you are downloading the FixPak image from the official IBM FTP site, make sure that the size of the downloaded image is the same as the one on the FTP site.

Note: The `installFixpak` script does not provide a trace option. `installAltFixPak` simply checks for the necessary pre-requisites and then calls the operating system's native install tool with certain customized options

Related reference:

- Applying the latest FixPak (Windows and UNIX)

License issues

Before you start debugging the license problem, you should have a better understanding of it, so you should ask yourself these questions:

1. What is the SQL error code? Check the *Message Reference* guide for further explanation and take the corrective action.
2. Is the license for a new installation? If this is a new installation, then you will need to verify that the DB2 license key matches with the version and product of DB2.
3. If this is not a new installation, has something else changed in the system? For example, perhaps there has been a change to the operating system (OS) or to file permissions, or perhaps a DB2 FixPak upgrade was installed.

4. Did you install the permanent DB2 license key? This is important because your DB2 server could be running in ″Try-and-Buy″ mode if the permanent license key has not been installed, and you cannot start DB2 because the ″Try-and-Buy″ mode has expired.

For example, let's say you receive the following SQL code

```
SQL8007W There are "90" day(s) left in the evaluation period for the product "DB2
Enterprise Server Edition". For evaluation license terms and conditions, refer to the
IBM Evaluation Agreement in the EVALUATE.AGR file, located in the following
directory: "C:\PROGRA~1\SQLLIB".
```

According to the Message Reference, SQL8007W indicates, ″`A valid license key has not been installed for this product. The evaluation period will expire after the specified number of days.`″

In this example, you would first make sure that you have installed the license key for ″DB2 Enterprise Server Edition.″ You could use the license management tool to do so. (The license management tool is discussed next.) If the license key is not installed, then you need to install it. Second, find out if there is any change made to the operating system.

Note: Finding out what has changed will give you a better idea as to what could have caused the DB2 licensing error.

Troubleshooting licensing problems

In order to verify that you have a valid DB2 license, use the license management tool called **db2licm**. To find out how to use it, run db2licm with the `-h` option to display help information.

Run ″`db2licm -l`″ on your test system to list all the DB2 products with license information. If this command does not return the appropriate license information, then this could mean either a valid DB2 license key is not installed or there is a potential permission problem with the nodelock file.

To confirm whether a valid DB2 license is installed, compare the license key inside the nodelock file with the license key file (for example, db2ese.lic is the license key file for DB2 UDB Enterprise Service Edition). Here are the different locations for the nodelock file based on the platforms:

- AIX - `/var/ifor`
- HP-UX, Linux and Solaris - `/var/lum`
- Windows - `\Program Files\SQLLIB\license`

Inside the nodelock file, comments line start with ′#′, and the license key is usually proceeded and followed by a comment line. For example,

```
#
<You will see the actual license key here.>
#[admin_comment] "IBM Toronto Lab" "DB2 Enterprise Edition" ...
```

If the license key is not yet installed, use db2licm to install it. If the license key does match the one on the product CD, check whether there is a permission problem with the nodelock file.

In UNIX, you should verify the file permission for the nodelock is set to `rw-r--r--` and owned by root.

Related reference:

- License management

Migration issues

This section will help you identify the source of common migration problems encountered when migrating DB2 Universal Database (UDB) Version 7 to DB2 UDB Version 8.

If you are migrating to DB2 UDB Version 8 from another database product, you should visit the DB2 UDB Migrate Now! Web site: www.ibm.com/software/data/db2/migration/

Related reference:

- Installation issues
- Migrating to DB2 UDB Version 8.2
- Overview of migrating to DB2 Information Integrator

Migration recommendations

Consider the following recommendations when planning your database migration:

Back up log files before migration when DB2® UDB uses replication

7 If you use replication for your DataJoiner® and DB2 UDB data, you must
7 archive all of the DB2 log files before migration.

7 For complete information on migrating your DB2 replication environment,
7 see the IBM® DB2 Information Integrator Migration Guide: Migrating to
7 SQL Replication Version 8 at
7 http://www.ibm.com/software/data/integration/db2ii/support.html.

DataJoiner instance migration

If you want to migrate an instance of DataJoiner or DB2 UDB on which you are running the Capture or Apply programs for DB2 replication, you must prepare to migrate your replication environment before you migrating the instance.

7 For complete information on migrating your DB2 replication environment,
7 see the IBM DB2 Information Integrator Migration Guide: Migrating to
7 SQL Replication Version 8 at
7 http://www.ibm.com/software/data/integration/db2ii/support.html.

Perform hardware and operating system upgrades separately from DB2 UDB migration

Performing hardware and operating system upgrades separately from DB2 migration simplifies problem determination if you encounter migration difficulties. If you upgrade your software or hardware prior to migrating to DB2, ensure that your system is operating acceptably before attempting DB2 migration.

7 **Dropping the detailed deadlocks event monitor**

7 At the same time a database is created, a detailed deadlocks event monitor
7 is also created. As with any monitor, there is some overhead associated
7 with this event monitor. If you do not want the detailed deadlocks event
7 monitor, then the event monitor can be dropped using the command:

```
7    DROP EVENT MONITOR db2detaildeadlock
```

7 To limit the amount of disk space that this event monitor consumes, the
7 event monitor deactivates, and a message is written to the administration
7 notification log, once it has reached its maximum number of output files.
7 Removing output files that are no longer needed allows the event monitor
7 to activate again on the next database activation.

Back-level server support

As you move your environment from DB2 Version 7 to DB2 Version 8, if you migrate your DB2 clients to Version 8 before you migrate all of your DB2 servers to Version 8, there are several restrictions and limitations. To avoid the known restrictions and limitations, migrate all of your DB2 servers to Version 8 before you migrate any of your DB2 clients to Version 8. These restrictions and limitations are not associated with DB2 Connect™; nor with zSeries®, OS/390®, or iSeries™ database servers.

For example:

- bring your DB2 UDB Version 7 clients up to the latest FixPak
- migrate your DB2 Servers to DB2 UDB Version 8 32-bit and then update to FixPak 8
- migrate your DB2 Connect server to Version 8 and then update to FixPak 8
- migrate your DB2 clients to DB2 UDB Version 8 and then update to FixPak 8
- migrate your DB2 UDB Version 8 servers to 64-bit

Benchmark DB2 performance

Run a number of test queries before migrating DB2. Record the exact environment conditions when queries are run. Also, keep a record of the **db2expln** command output for each test query. Compare the results before and after migration. This practice can help to identify and correct any performance degradation.

Devise a migration planning document

You should create a migration planning document. Consider using this sample document as a template: http://d02dbs88.southbury.ibm.com/support/swg/dmgtech.nsf/f2bc0e4fed485c5f48256bb

Devise a plan to back out of a migration

There is no utility to reverse a migration. If you must back out of a migration, you might have to remove DB2 Version 8 code from your system, reinstall the previous version of DB2 to recreate back-level instances, and restore your database backups. If you have to back out of a migration, current database backups and detailed records of database and database configuration settings are essential.

Migrating instances with DB2 DataPropagator™ replication

Before migrating an instance of DataJoiner or DB2 UDB on which you are running the Capture or Apply programs for DB2 DataPropagator, read the migration documentation for DB2 DataPropagator Version 8. You must prepare to migrate your replication environment before you migrate the DB2 or DataJoiner instance. You must also perform specific tasks immediately after the migration of your DB2 or DataJoiner instance. Migration documentation for DB2 DataPropagator Version 8 can be found at the http://www.ibm.com/software/data/dpropr/library.html Web site.

Related concepts:

- Benchmark testing

- Explain tools

Related tasks:

- Migrating DB2 UDB (Windows)
- Migrating DB2 UDB (UNIX

Related reference:

- DB2 Universal Database planned incompatibilities
- Version 8 incompatibilities with previous releases
- Version 7 incompatibilities with previous releases

Migration restrictions

You should be aware of the following restrictions before you migrate to DB2 UDB Version 8:

- Migration is supported only from:
  - DB2 UDB Version 6.x or Version 7.x. (All platforms supported in Version 6.x and Version 7.x; Linux must be at Version 6 FixPak 2.)
  - DB2 DataJoiner V2.1.1 32-bit (AIX, Windows NT, and Solaris Operating Environment).
- Issuing the **migrate database** command from a DB2 UDB Version 8 client to migrate a database to a DB2 Version 8 server is supported; however, issuing the migration command from an DB2 UDB Version 6 or Version 7 client to migrate a database to a DB2 UDB Version 8 server is not supported.
- When migrating from DB2 DataJoiner V2.1.1, DB2 Information Integrator is required to support non-IBM data sources.
- Migration between platforms is not supported. For example, you cannot migrate a database from a DB2 server on Windows to a DB2 server on UNIX.
- Migrating a partitioned database system that has multiple computers requires that database migration be performed after DB2 UDB Version 8 is installed on all participating computers. Any DB2 migration commands need to be run on each of the participating computers.
- Windows allows only one version of DB2 UDB to be installed on a computer. For example, if you have DB2 UDB Version 7 installed and install DB2 UDB Version 8, DB2 UDB Version 7 is removed during the installation. All instances are migrated during DB2 installation on Windows operating systems.
- User objects within your database cannot have DB2 UDB Version 8 reserved schema names as object qualifiers. These reserved schema names include: SYSCAT, SYSSTAT, and SYSFUN.
- User-defined distinct types using the names BIGINT, REAL, DATALINK, or REFERENCE must be renamed before migrating the database.
- You cannot migrate a database that is in one of the following states:
  - Backup pending
  - Roll-forward pending
  - 7 One or more table spaces in an abnormal state
  - Transaction inconsistent
- Restoration of back-level (DB2 Version 6.x or Version 7.x) database backups is supported, but rolling forward of back-level logs is not supported.
- Database transactions executed between database backup time and the time DB2 UDB Version 8 migration is completed are not recoverable.

Reverse FixPak upgrade restrictions

Beginning with DB2 UDB Version 8 FixPak 1, the maximum number of tablespaces supported in a database increases from 4096 to 32768.

For the most part, you should be able to move your database from DB2 UDB Version 8 to DB2 UDB Version 8 FixPak 1 without noticing a change or having to do anything special to use the new tablespace limit.

Note the following restrictions related to moving a database from DB2 UDB Version 8 FixPak 1 (or a later DB2 UDB Version 8 FixPak) back to the DB2 Version 8 level:

- If you want to move from DB2 Version 8.2 back to DB2 Version 8.1, you will need to run the **db2demigdbd** command before going back to DB2 Version 8.1. The **db2demigdbd** is a reverse database directory files tool that restores your database directory to its Version 8.1 format.
- Moving a database that contains a tablespace ID higher than 4096 to DB2 Version 8 from DB2 Version 8 FixPak 1 or later is unsupported. Attempting to do so will result in anomalous behavior and improper operation.
- Restoring a database image that contains a tablespace ID higher than 4096 on DB2 Version 8 is unsupported. Attempting to do so results in anomalous behavior and improper operation.
- When moving from DB2 UDB Version 8 FixPak 1 (or a later DB2 UDB Version 8 FixPak) back to DB2 UDB Version 8, the log-skipping functionality is disabled until such time as the DB2TSCHG.HIS file is removed.

Migrating databases

Prerequisites:

You require SYSADM authority.

Restrictions:

Migration is supported only from:

- DB2 Version 6.x or Version 7.x. (all platforms supported in Version 6.x and Version 7.x)
- DB2 DataJoiner Version 2.1.1 (AIX, Windows NT, and Solaris Operating Environment).

No database migration is required if the database has been migrated to a DB2 Version 8 FixPak level.

Procedure:

To migrate a DB2 database:

1. Migrate the database using the **db2 migrate database** command.

DB2 MIGRATE DATABASE command

```
►►──MIGRATE──┬─DATABASE─┬──database-alias───────────────────────────────►
             └─DB───────┘
```

```
►─┬────────────────────────────────────┬──────────────────────────────►◄
  └─USER─username─┬──────────────────┬─┘
                  └─USING─password───┘
```

where:

DATABASE *database-alias*
Specifies the alias of the database to be migrated to the currently installed version of the database manager.

USER *username*
Identifies the user name under which the database is to be migrated.

USING *password*
The password used to authenticate the user name. If the password is omitted, but a user name was specified, the user is prompted to enter it.

7 2. Optional: Update statistics for local tables within the database. When database migration is complete, old statistics that are used to optimize query performance are retained in the catalogs. However, DB2 Version 8 has statistics that are modified or do not exist in DB2 Version 6 or DB2 Version 7. To take advantage of these statistics, you may want to execute the **runstats** command on tables, particularly those tables that are critical to the performance of your SQL queries.
3. Optional: Rebind packages. During database migration, all existing packages are invalidated. After the migration process, each package is rebuilt when it is used for the first time by the DB2 Version 8 database manager. You can run the **db2rbind** command to rebuild all packages stored in the database.
4. Optional: Revoke EXECUTE privileges on external stored procedures that contain SQL data access from PUBLIC. During database migration, EXECUTE privileges are granted to PUBLIC for all existing functions, methods, and external stored procedures. This will cause a security exposure for external stored procedures that contain SQL data access which allow users to access SQL objects for which they would not otherwise have privileges. Revoke the privileges by entering the **db2undgp - r** command.
5. Optional: Migrate DB2 Explain tables.
6. Optional: If you recorded configuration settings before migration, you might want to compare pre-migration configuration settings to current configuration settings to verify successful migration. Verify:
   - database configuration parameter settings
   - database manger configuration parameter settings
   - tablespaces records
   - packages records

Note: During migration, the database configuration parameter *maxappls* is set to automatic. If you want it set to a different value, you should update it manually.

Application Development issues

Applications or stored procedures are written in a variety of languages using a set of standard APIs to manipulate database data.

It is important to know the layers involved in DB2 applications in order to diagnose issues. The DB2 Call Level Interface (CLI), which is equivalent in functionality to ODBC, is often used underneath other drivers, so it may be necessary to debug different layers to determine problems. The figure below shows the typical application flow:

Figure 2.

By following the *DB2 Application Development* manuals as well as the specific documentation related to your programming interface, you can avoid most errors involved in creating and executing DB2 applications. DB2 product manuals for application development can be found in both HTML and PDF formats at http://www-306.ibm.com/software/data/db2/udb/support/manualsv8.html .

Related references:

- DB2 Supported Programming Interfaces
- DB2 supported development software

Debugging and Optimizing an Application

You can debug and optimize your application while you develop it.

Procedure:

To debug and optimize your application:

- Prototype your SQL statements. You can use the command line processor, the Explain facility, analyze the system catalog views for information about the tables and databases that your program is manipulating, and update certain system catalog statistics to simulate production conditions.
- Use the flagger facility to check the syntax of SQL statements in applications being developed for DB2 Universal Database for z/OS and OS/390, or for conformance to the SQL92 Entry Level standard. This facility is invoked during precompilation.
- Make full use of the error-handling APIs. For example, you can use error-handling APIs to print all messages during the testing phase.
- Use the database system monitor to capture certain optimizing information for analysis.

Proper application error handling

Often applications are difficult to troubleshoot because not enough error information is provided by the application.

The method used to retrieve additional error information is different depending on the type of application, Instructions for how to devise appropriate error handling can be found in the Application Development documentation, in particular in the sections listed in the Related Reference below.

Example 1: Adding error handling calls to a simple CLI application

Sample1.c is a simple CLI application that has a problem upon execution.

When the application is run, it uses the current user ID and password to connect with a database named `SAMPLE`. It allocates the appropriate environment and database handles, then connects to the database.

Thereafter it performs a fetch as follows:

```
sqlrc = SQLAllocHandle( SQL_HANDLE_STMT, hdbc, &hstmt ) ;
    if ( sqlrc != SQL_SUCCESS )
    {
        printf("\n Error allocating statement handle.\n");
        return( 1 ) ;
    }

    sqlrc = SQLFetch( hstmt ) ;
    if ( sqlrc != SQL_SUCCESS )
    {
        printf("\n Error Fetching.\n");
        return( 1 ) ;
    }

    sqlrc = SQLFreeHandle( SQL_HANDLE_STMT, hstmt );
    if ( sqlrc != SQL_SUCCESS )
    {
        printf("\n Error freeing statement handle.\n");
        return( 1 ) ;
    }
```

When the program is executed, the following error occurs:

```
Error Fetching.
```

The first step in diagnosing this problem is to examine the code, in order to find where this error is generated. In this case, you would find that the SQLFetch call must have returned an SQL_ERROR. Since the application didn't request the error details, it is difficult to determine the cause. You could obtain a CLI trace (see ″Enabling CLI traces″)and look for the line which includes the SQLFetch call. It would be similar to the following:

```
SQLFetch( hStmt=1:1 )
---> Time elapsed - +2.200000E-004 seconds
SQLFetch( )
<--- SQL_ERROR Time elapsed - +1.238000E-003
seconds
```

This shows the error being returned, but no error details. In many cases, detailed diagnostic information is returned only when an application requests it. Taking a CLI trace may reveal more information than the application returns.

If you were to rewrite this sample program to gather proper error information, you would need to add calls to the function SQLGetDiagRec. For example, you could create a simple procedure called GetCLIErrorInfo which calls SQLGetDiagRec and which is in turn called after the fetch.

```
...
void GetCLIErrorInfo( SQLSMALLINT htype, /* handle type identifier */
                             SQLHANDLE   hndl  /* handle */                          )
{
...
    while ( SQLGetDiagRec( htype,
                           hndl,
                           i,
                           sqlstate,
                           &sqlcode,
                           message,
                           SQL_MAX_MESSAGE_LENGTH + 1,
                           &length
                         ) == SQL_SUCCESS ) {
       printf( "\n  SQLSTATE         = %s\n", sqlstate ) ;
       printf( "  Native Error Code = %ld\n", sqlcode ) ;
       printf( "%s\n", message ) ;
       i++ ;}
...}

int main()
...
sqlrc = SQLFetch( hstmt ) ;
    if ( sqlrc != SQL_SUCCESS )
    {
        printf("/n Error Fetching./n");
        GetCLIErrorInfo( SQL_HANDLE_STMT, hstmt );
    }
 ...
```

Once such error handling has been added, the sample program would return output as follows:

```
Error Fetching.
SQLSTATE = HY010
Native Error Code = -99999
[IBM][CLI Driver] CLI0125E Function sequence error.SQLSTATE=HY010
-------------------------
Application Complete.
```

By checking the ODBC Standard or DB2 CLI documentation, you will learn that a function sequence error is returned from an SQLFetch call if you do not call `SQLExecute` first. The statement needs to be executed before you can fetch any data it may return. This is a simple example of a situation that can be resolved only after appropriate error handling has been added to the application.

Related Reference:

- Diagnostics in CLI applications overview
- Error-Handling Considerations in Partitioned Database Environments
- Error Message Retrieval in an Application
- Handling an SQLException under the DB2 JDBC Type 2 Driver
- Handling an SQLException under the DB2 Universal JDBC Driver
- Handling SQL errors in an SQLJ application
- Returning error messages from SQL procedures

Displaying the contents of a bind file using the db2bfd tool

There are times when it is beneficial to examine the contents of a DB2 bind file. For example, a vendor might supply an executable and bind file and you might want to see what kinds of SQL statements the application might execute. Also, consider the case where an errant application is deleting data from a table that is supposed to remain read-only but you aren't sure which application is responsible. By examining the bind files for all of the applications that run against the database, you might be able to find a DELETE statement against the table in question and from there dig further into the application's logic to determine what is going wrong.

The **db2bfd** tool can be used to display these kinds of thing as well as some other information.

Execute the tool without any parameters to see its usage.

```
C:\>db2bfd

Usage: db2bfd [ [-b] [-h] [-s] [-v] ] <filespec>

Where: <filespec> is a V7 or V8 bind file
```

```
Options: -b = display bind file header
-h = display this information
-s = display SQL statements
-v = display host variable declarations
```

Example 1: Use `db2bfd -s` to see the SQL statements contained in the db2sampl.bnd file

On Windows® operating systems:

```
db2bfd -s %DB2PATH%\bnd\db2sampl.bnd
```

(where `%DB2PATH%` is a variable that determines where DB2® is installed)

On UNIX®:

```
db2bfd -s $HOME/sqllib/bnd/db2sampl.bnd
```

(where `$HOME` is the home directory of the instance owner)

Next, use the -v option to see the corresponding host variables that would have been defined in the source code for db2sampl:

On Windows® operating systems:

```
db2bfd -v %DB2PATH%\bnd\db2sampl.bnd
```

(where `%DB2PATH%` is a variable that determines where DB2® is installed)

On UNIX®:

```
db2bfd -v $HOME/sqllib/bnd/db2sampl.bnd
```

(where `$HOME` is the home directory of the instance owner)

Related reference:

- db2bfd - Bind File Description Tool Command
- Package Creation Using the BIND Command

Resolving compilation errors

Errors that occur during compilation are often due to programming errors or environments that are not correctly set up for application creation.

Here are some basic steps to ensure that your environment is setup correctly:

- Ensure makefiles contain the correct compiler information
- Set all compiler-related variables for include files and libraries
- Ensure any command line arguments point to the correct DB2 directories

Here are some of the common mistakes that can lead to linker errors:

- Improper API/function calls
- Multiple versions of linked libraries in the LIBPATH
- API/function definitions not matching the code contained in the library files

Refer to the samples included with DB2, as they include scripts and instructions to compile and link the sample applications.

Example 1: Investigating a compilation error in a simple CLI application written in C/C++

When attempting to compile an application (named test.c), the following command was used:

```
cl test.c -I "%DB2PATH%\include" -link"%DB2PATH%\lib\db2cli.lib"
```

The following error was encountered:

```
sample.c
sample.c(19) : warning C4047: 'function' : 'long ' differs in levels of indirect
ion from 'long *'
sample.c(19) : warning C4024: 'SQLAllocHandle' : different types for formal and
actual parameter 2
sample.c(19) : error C2198: 'SQLAllocHandle' : too few actual parameters
```

The first step in problem determination is to check the line of code that the compiler objects to. In this case, it is as follows:

```
    sqlrc = SQLAllocHandle( SQL_HANDLE_ENV, &henv ) ;
    if ( sqlrc != SQL_SUCCESS )
    {
        printf("/n Error allocating environment handle./n");
        return -1;
    }
```

Since the error indicated that there were too few parameters, you would then check the CLI documentation for SQLAllocHandle. You would see that the call is indeed incorrect. It is missing one input handle. You can find the SQLAllocHandle call documented at http://publib.boulder.ibm.com/infocenter/db2help/topic/com.ibm.db2.udb.doc/ad/r0000556.htm

This is a simple error common to all types of programming. Errors such as this, received at compile time, can usually be resolved by checking the related documentation and correcting the code as necessary.

Related reference:

- DB2 supported development software

Avoiding linker errors

Compiler and linker errors for DB2 applications are often the same as those you encounter when creating any type of application. The linking step sometimes results in difficulty creating the executable.

Ensure your environment is correctly installed and you are using the proper DB2 static and shared libraries (e.g. those contained in sqllib/lib32 or sqllib/lib). This is the best way to avoid compilation/linking problems. You should check the *DB2 Application Development* guides relating to your programming interface and your compiler specific documentation. They will guide you to set up the correct environment.

Here are some of the common items that relate to having your environment setup correctly:

- Ensure makefiles contain the correct compiler information
- Set all compiler-related variables for include files and libraries
- Ensure any command line arguments point to the correct DB2 directories

Here are some of the common mistakes that can lead to linker errors:

- Improper API/function calls
- Multiple versions of linked libraries in the LIBPATH
- API/function definitions not matching the code contained in the library files

You can look at the samples included with DB2 as they also include scripts and instructions to compile and link them.

Troubleshooting suspended or looping applications

It is possible that, after you start a query or application, you suspect that it is suspended (it does not show any activity) or that it is looping (it shows activity, but no results are returned to the application). Ensure that you have turned lock timeouts on. In some situations, however, no error is returned. In these situations, you may find the diagnostic tools provided with DB2® and the database system monitor snapshot helpful.

One of the functions of the database system monitor that is useful for debugging applications is to display the status of all active agents. To obtain the greatest use from a snapshot, ensure that statement collection is being done before you run the application (preferably immediately after you run DB2START) as follows:

```
db2_all "db2 UPDATE MONITOR SWITCHES USING STATEMENT ON"
```

When you suspect that your application or query is either stalled or looping, issue the following command:

```
db2_all "db2 GET SNAPSHOT FOR AGENTS ON database
```

Error-Handling Considerations in partitioned database environments

In a partitioned database environment, DB2® breaks up SQL statements into subsections, each of which is processed on the partition that contains the relevant data. As a result, an error may occur on a partition that does not have access to the application. This condition does not occur in a nonpartitioned database environment.

You should consider the following:
- Non-CURSOR (EXECUTE) non-severe errors
- CURSOR non-severe errors
- Severe errors
- Merged multiple SQLCA structures
- How to identify the partition that returned the error

If an application ends abnormally because of a severe error, indoubt transactions may be left in the database. (An indoubt transaction pertains to global transactions when one phase completes successfully, but the system fails before the subsequent phase can complete, leaving the database in an inconsistent state.)

DB2 Development Center issues

Choose one of the following:

- Enabling the Next push button in a wizard
- Correcting a user ID or password
- Resolving routines that fail to build on another connection
- Supporting null input values to Java stored procedures

- Testing null input values to SQL user-defined functions
- Diagnosing SQL routines that fail to build
- Correcting color schemes that make text unreadable
- Diagnosing SQL routine build failures with the DSNTPSMP build utility
- Diagnosing Java routine build failures with the DSNTJSPP build utility
- Diagnosing routine build failures on DB2 UDB for OS/390 or DB2 UDB for z/OS servers, error code -471
- Setting display for the pop-up information for fields and controls
- Diagnosing system connection exceptions when trying to establish multiple JDBC connections (AIX)
- Maintaining correct formatting for stored procedure source code (DB2 for iSeries)

Enabling the Next push button in a wizard

Symptom:

When using a wizard, the Next push button is not available after you type text in the text fields.

Possible cause:

The text that you enter in each fieldmust conform to the rules for that field. The Next pushbutton is not available until the text that you enter conforms to the rules.If you did not receive notification that your text entries do not follow therules, turn on the constraint checking option.

Action:

- Close the wizard.
- Open the Environment Settings notebook.
- On the User-assistance features page, select all check boxes in the Constraintchecking group to turn on the constraint checking options.

Correcting a user ID or password

Symptom:

The user ID or password that you typed to connect to the database server is incorrect.

Possible causes:

The authentication configuration of the database server is set to SERVER, and your user ID and password are not valid on the server. This is the default configuration for DB2 Universal Database.

The authentication configuration of the database server is set to CLIENT, and your user ID and password are not valid on the client workstation.

Action:

Check the authentication configuration of the database to which you are trying to connect. If the configuration is set to SERVER, your user ID and password must be valid values on the server. If the configuration is set to CLIENT, your user ID and password must be valid values on the client workstation.

Resolving routines that fail to build on another connection

Symptom:

When you copy a routine to another connection, the build fails.

Possible causes:

Building the routine to another database could have failed because of the following reasons:

- The routine contained references to objects (such as schemas and tables) that do not exist on the target database.
- The SPECIFIC NAME of the object being pasted already exists in the target database.

Actions:

The following actions may resolve the symptom:

- Check the source code of the routine and remove references to objects that do not exist on the target database, or create the objects on the target database.
- Change the specific name by editing the SPECIFIC clause of the SQL procedure or UDF, or editing the Specific Name property of a Java stored procedure.

Supporting null input values to Java stored procedures

Symptom:

A null value is passed to a Java stored procedure input parameter that was declared with a Java primitive type (for example, an int). DB2 generates an SQL error message without calling the stored procedure.

Possible cause:

On a workstation server, passing null values to input parameters is always allowed. On DB2 for OS/390 or z/OS, passing null values to input parameters is allowed if you specify SIMPLE WITH NULLS for the linkage. The Insert Java Stored Procedure wizard declared an input parameter using a Java primitive type. DB2 generates an SQL error message because the input parameter that was declared with the primitive type was passed a null value.

Action:

Change the primitive type to the corresponding Java class in the Development Center source code editor to allow the stored procedure to support null input values. For example, the Java primitive type int becomes Integer.

Testing null input values to SQL user-defined functions

Symptom:

A null value passed to a SQL user-defined function input parameter generates an SQL error message without calling the UDF.

Possible cause:

There is a known DB2 restriction that prevents Java applications, including Development Center, from sending a null value to input parameters of a user-defined function.

Action:

To test SQL user-defined functions that accept NULL values, call them from a DB2 Command Processor session: `db2 ==> values(<udfname>(<parameter values,including NULL,separated by commas>))`

Diagnosing SQL routines that fail to build

Symptom:

Building an SQL routine results in the error message ″Build not successful: -1.″ No other error messages are shown.

On Windows, the user may get an error messages such as: ’`nmake`’ `is not recognized as an internal or external command, operable program or batch file`.

Possible causes:

With DB2 versions prior to 8.2, a C compiler is required to build SQL stored procedures, so possible causes include:

- A C compiler is not installed on the database server.
- A C compiler is installed on the database server but is not properly configured. For example, environment variables required by the C compiler may be incorrect.

Actions:

For database servers on Windows NT, 2000, or XP:

- Ensure that Visual C++ Version 6 is installed.
- Ensure that the server’s DB2_SQLROUTINE_COMPILER_PATH is set to the location of the file vcvars32.bat.
- Optional: Move the Visual C++ environment settings for PATH, INCLUDE, and LIB from User settings to System settings.

Additional configuration information is available in the *IBM DB2 UDB Application Development Guide in the Building and Running Applications and Setting Up the Application Development Environment topics*.

Correcting color schemes that make text unreadable

Symptom:

You cannot read the text in the editor because the text and background are the same or nearly the same color.

Possible cause:

Using the Environment Settings notebook, you selected a foreground and background color that are the same or almost the same.

Actions:

- If you did not close the Environment Settings notebook or apply the color changes, you can apply the previous color selection by clicking Reset on the Editor page of the Environment Settings notebook.
- If you closed the Environment Settings notebook or applied the color changes, on the Editor page of the Environment Settings notebook, choose a different color scheme by clicking Change. The Change Colors window opens.

Diagnosing SQL routine build failures with the DSNTPSMP build utility

Symptom:

When you try to build an SQL object to an OS/390 database server, the build fails and returns the following error message:

```
DSNTPSMP - returned nnn
Rolling back...successful
procedure-name - Build failed.
```

Where `procedure-name` is the name of the routine. nnn is a numeric error code. See the Message Reference for your operating system for an explanation of error codes.

Possible cause:

There might be a problem with the SQL object source code.

Actions:

1. Review the diagnostic information to determine why the build failed. The DSNTPSMP build utility might return a result set of messages that describe where and why the SQL object build failed.
2. Correct the problem that is described by the diagnostic information.
3. Rebuild the SQL object.
4. If additional information is needed for diagnosing the problem look at the SYSLOG for the WLM procedure where DSNTPSMP executes. Often, a dataset may fill up and be the cause for the build failure. This can be determined by looking directly at the WLM procedure output

Diagnosing Java routine build failures with the DSNTJSPP build utility

Symptom:

When you try to build an SQL object to an OS/390 database server, the build fails and returns the following error message:

```
DSNTJSPP - returned nnn
Rolling back...sucessful
procedure-name - Build failed
```

Where nnn is a numeric error code. See the DB2 Universal Database for OS/390 and z/OS Messages and Codes book (GC26-9940) for an explanation of error codes.

Possible cause:

There might be a problem with the SQL object source code.

Actions:

1. Review the diagnostic information to determine why the build failed. Obtain available diagnostic information by selecting the Verbose build option when using the wizard, or from the Stored Procedure Properties notebook. Selecting this option causes DSNTJSPP to return a result set of messages to Development Center that describe where and why the SQL object build failed.
2. Correct the problem that is described by the diagnostic information.
3. Rebuild the Java object.
4. If additional information is needed for diagnosing the problem, look at the SYSLOG for the WLM procedure where DSNTJSPP and DSNTBIND execute.

Diagnosing routine build failures on DB2 UDB for OS/390 or DB2 UDB for z/OS servers, error code -471

Symptom:

When you try to build an SQL or Java object to an OS/390 or z/OS database server, the build fails and you receive the following error message:

```
Create object returns -471
[IBM][CLI Driver][DB2] SQL0969N
There is no message text corresponding to SQL error "-471" in the message file on this workstation
The error was returned from module "DSNX9WCA" with original tokens "DSNTPSMP 00E79002 ".
SQLSTATE=55023
```

Possible causes:

The Work Load Manager (WLM) was not started.

Actions:

1. Ensure that the SQL objects build utility DSNTPSMP is properly installed on your DB2 for OS/390 or z/OS server.
2. Ensure that the Java objects build utility DSNTJSPP and DSNTBIND are properly installed on your DB2 for OS/390 or z/OS server.

Contact your system administrator for information about obtaining the build utility.

Setting display for the pop-up information for fields and controls

Symptom:

In the Development Center wizards and notebooks, the pop-up information that explains the fields and controls does not display.

Possible causes:

- The pop-up information windows are turned off.
- You are not holding the mouse pointer over a field or control long enough for the pop-up information to appear.
- You are not pressing the F1 key when a field or control is in focus.

Actions:

- Ensure that the pop-up information windows are not turned off:
  1. Open the Environment Settings notebook.
  2. On the User-assistance features page, ensure that the Show pop-up information for interface controls check box is selected.
  3. In the Delay pop-up information in milliseconds field, type the time delay in milliseconds. The value milliseconds is an integer. For example, entering 1000 as the value results in a 1000 millisecond (1 second) delay before pop-up information appears.
- Press the F1 key on your keyboard when the field or control is in highlighted. To move the highlight from one field or control to another, press the Tab key. If no field or control is highlighted, the browser help displays.

Diagnosing system connection exceptions when trying to establish multiple JDBC connections (AIX)

Symptom:

You have trouble viewing objects in the Server View and experience system connection exceptions such as:

```
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] SQL1224N
A database agent could not be started to service a request, or was terminated
as a result of a database system shutdown or a force command.
SQLSTATE=55032
```

Possible cause:

The number of JDBC connections is limited by the number of shared memory segments to which a single process can be attached. By default, AIX does not permit 32-bit applications to attach to more than 11 shared memory segments per process, of which a maximum of 10 can be used for local DB2 connections. Development center requires multiple JDBC connections. The EXTSHM environment variable must be configured for multiple JDBC connections.

Action:

To resolve this issue if you are using AIX version 4.3.1 or newer, set set the environment variable EXTSHM to ON to increase the number of shared memory segments to which a single process can be attached. EXTSHM must be exported both in the shell where the client application is started and also in the shell where db2start is run. To configure the EXTSHM environment variable for multiple JDBC connections:

1. In client sessions, before starting the client application, type the following command: `export EXTSHM=ON`
2. Before starting the DB2 server, type the following commands:

   ```
   export EXTSHM=ON
   db2set DB2ENVLIST=EXTSHM
   db2set -all
   ```

3. Add the following lines to the sql/db2profile:

```
EXTSHM=ON
export EXTSHM
```

Maintaining correct formatting for stored procedure source code (DB2 for iSeries)

Symptom:

On iSeries, the source code of a stored procedure body loses all its formatting and appears on a single line.

Possible cause:

You created the stored procedure using:
- STRSQL
- RUNSQLSTM command
- CREATE PROCEDURE statement in embedded SQL

Stored procedures created using one of these methods will lose the formatting of their bodies in Development Center.

Action:

If you want to make the stored procedure appear formatted correctly, you must recreate the procedure using:
- Development Center
- the Run SQL Scripts interface in the iSeries Navigator
- the New External Procedure or New SQL Procedure interface in the iSeries Navigator

Backup and recovery issues

Taking frequent database backups is essential for the recovery of databases in the event of a disaster. Understanding the recovery options and high level processing of recovery can reduce the anxiety that is commonly associated with recovery. Database recovery is one of the crucial tasks that is performed by the database administrator. Refer to Developing a backup and recovery strategy.

The main backup and restore processes are:

db2agent
co-ordinator of all child processes and handles all communications with the application.

db2bm
db2 buffer manipulator - uses prefetchers to process I/O requests. Places data into or reads data from the backup restore buffers. Passes data to the db2med process.

db2med
db2 media controllers - performs I/O with backup destination . Data is written from or read into the backup / restore buffers. Buffers passed back to db2bm.

Verifying backup images using the db2ckbkp command

DB2 UDB provides a utility called **db2ckbkp** that can be used to validate a backup image. For details about the options, run db2ckbkp with no arguments.

Below is an example scenario in which the db2ckbkp utility may be used.

```
SQL1013N The database alias name or database name "ITB20" could not be
found.
2004-09-04-01.48.22.938044 Instance:db2inst1 Node:000
PID:1081544(db2agent (ITB20) 0) TID:1 Appid:*LOCAL.DB2INST1.021012172526
database_utilities sqludMRWarn Probe:40 Database:ITB20
DiagData
.
16f6 ffff 433a 5c44 4232 4261 636b 7570 . C:\DB2Backup
5c49 5442 3230 2e30 5c44 4232 5c4e 4f44 \ITB20.0\DB2\NOD
4530 3030 305c 4341 544e 3030 3030 5c32 E0000\CATN0000\2
3030 3230 3931 335c 3039 3237 3339 2e30 0020913\092739.0
3031 00 01.
```

The above error indicates that the backup image cannot be found. It is possible that the image exists in that specified location but may be corrupted. To verify that the backup image is good, run db2ckbkp with the -a option to dump as much information as possible.

If there is no problem with the backup image the following message will be returned:

```
Image Verification Complete - successful.
```

Related reference:

- db2ckbkp - Check Backup

History file analysis

Every DB2 UDB database has a history file that records administrative operations. A recovery history file is created with each database and is automatically updated. During a database migration, the history file is migrated as well. The history file can be accessed by issuing the following command:

```
db2 list history all for <dbname>
```

The database history file is invaluable in a recovery scenario. The history file is individually restorable from any backup image. If the current database is unusable or not available, and the associated recovery history file is damaged or deleted, an option on the **RESTORE** command allows only the recovery history file to be restored. The recovery history file can then be reviewed to provide information on which backup to use to restore the database. For example, you can restore the history file for the sample database with this command:

```
db2 restore database sample history file
```

The recovery history file provides enough information to recover a database or table space using a backup image.

```
Extract Of A History File
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-- --- ------------------ ---- --- ------------ ------------ --------------
B D 20050314154131001 F D
----------------------------------------------------------------------------
Contains 2 tablespace(s):
00001 SYSCATSPACE
00002 USERSPACE1
----------------------------------------------------------------------------
```

```
Comment: DB2 BACKUP SAMPLE ONLINE
Start Time: 20050314154131
End Time:
----------------------------------------------------------------------------
EID: 4 Location: C:\SAMPLE.0\DB2\NODE0000\CATN0000\20050314
```

Some of the important information in the history file is

Op This is the operation that was performed. In the exampled above it is a ″B″ which stands for backup. A list of the possible values for **Op** can be found in the Command Reference under the `LIST HISTORY` command.

Obj This is the granularity of the backup. ″D″ for database backup . ″T″ for tablespace backup.

Earliest Log
In the case of an online backup, this is the first log required for the rollforward operation.

Current Log
The last log that was written to when the backup completed. In the case of an online backup, this is the minimum log required for the backup to complete.

Related reference:

- Understanding the recovery history file

DB2 Data Links Manager issues

1 Setting up a backup strategy for DB2 Data Links Manager

1 The following system setup and backup procedures are recommended for easier
1 recovery.

1 1. Place the Data Links File Manager (DLFM) database (usually called DLFM_DB),
1 any file systems under control of the Data Links Filesystem Filter (DLFF), the
1 server archive area, and the DLFM home directory on different file systems.
1 Ensure they do not share disks.

1 2. Back up any file systems under the control of the DLFF and the DLFM_DB on
1 a regular basis. In the event of disk failure, this backup will provide added
1 recoverability.

1 Many commercial backup-and-restore programs are available for all types of
1 file systems. These programs enable you to back up your file systems onto any
1 media that you choose, and typically provide easy-to-use restore operations. An
1 example of such a program is IBM's Tivoli Storage Manager. Tivoli Storage
1 Manager runs in all environments that the DB2 database product supports.

1 3. Back up the archive area or archive server on a regular basis, if possible.

1 The archive area itself is used as a repository for DB2 Data Links
1 Manager-initiated backups of DLFM data, and possibly linked files. Keeping
1 these backups might result in some data redundancy, but having them will
1 provide added recoverability in the event of disk failure.

1 4. Attempt to have only **one** DB2 database associated with **one or more** DB2 Data
1 Links Managers. If possible, avoid having multiple databases associated with
1 one DB2 Data Links Manager, because certain recovery scenarios will become
1 more complex than is necessary. Perform full database backups of the DB2
1 databases and the DLFM database (DLFM_DB) on the Data Links server at
1 regular intervals. Coordinate the timing of the database backups so that they
1 occur together, and ensure that their data is transactionally consistent.

An alternative approach to backing up a Journaled File System on AIX systems

This section describes a high availability backup method for an AIX Journaled File System environment. Using this approach enables you to perform a file system backup without stopping your DB2 Data Links Manager.

This backup method uses the following programs:

- `online.sh`
- `quiesce.c`

These programs are provided with your DB2 Data Links Manager installation software and are located under the `/sqllib/samples/dlfm` directory. The `online.sh` script, which calls the `quiesce.c` program, performs the following actions:

- Temporarily deactivates all the tables in databases that are registered with the DB2 Data Links Manager. This stops any new Data Links Manager activity.
- Unmounts and remounts the file system as a read-only file system.
- Performs a file system backup.
- Unmounts and remounts the file system as a read-write file system.
- Resets and reactivates the database tables.

Prerequisites:

To use the `online.sh` script, you must have a catalog entry on the DB2 Data Links Manager node for each database that is registered with the DB2 Data Links Manager. You must also have the complete entry for the Data Links File System (DLFS) on the `/etc/filesystems` file.

Procedure:

To execute the backup, perform the following steps:

1. Copy the `quiesce.c` CLI source file and the `online.sh` shell script to directory of your choice on the Data Links server where the DLFS is located.

   You will modify and work with these copies in the rest of this procedure.
2. Compile `quiesce.c` using the following command:

   ```
   xlC -o quiesce -ldb2 -L$HOME/sqllib/lib -I$HOME/sqllib/include quiesce.c
   ```
3. Modify the `online.sh` script to suit your environment:
   a. Select a backup command for the do_backup function of the `online.sh` script. At the top of the do_backup function, there are several commented lines of backup options. Remove the comment characters from the lines you want to use, and insert any necessary backup commands and parameters.
   b. At the top of the script, replace the specified default DLFM_INST environment variable with your Data Links File Manager instance name.
   c. On the next line, replace the default PATH_OF_EXEC environment variable with the path where your `quiesce.c` executable resides.
4. From the Data Links server where the DLFS is located, run the script as follows:

   ```
   online.sh <filesystem_name>
   ```

File system backup and restore utilities

To ensure your IT environment's data integrity and recoverability, it is imperative that you institute a regular backup scheme.

Many commercial backup-and-restore programs are available for all types of file systems. These programs enable you to back up your file systems onto any media that you choose, and typically provide easy-to-use restore operations. An example of such a program is IBM's Tivoli Storage Manager. Tivoli Storage Manager runs in all environments that the DB2 database product supports.

Your operating system will also provide backup-and-restore utilities. Windows NT and Windows 2000 provide the Backup utility. AIX provides the System Storage Management utilities. On Solaris operating environments, you can use the **ufsdump** and **ufsrestore** commands. See your operating system documentation for more information about the backup utilities.

Important: To reduce recovery time after a crash, use an incremental backup strategy where level 0 refers to a full backup, and levels 1 through 9 refer to incremental backups. A level `n` backup backs up only those files that have changed since a level `(n-1)` backup. After a level `n` backup, the next backup to be taken will be a level `(n+1)` backup.

1 Obtaining Data Links Manager environment information using
1 the dump utility (dlfm_dump)

1 When you report problems to IBM Service, you might be asked to run the Data
1 Links File Manager (DLFM) dump utility, `dlfm_dump`, to provide the service
1 personnel with details of your DB2 Data Links Manager environment.

1 The dlfm_dump utility enables you to obtain a "snapshot" of significant data that
1 is stored in the Data Links File Manager's own database (called `DLFM_DB` by default
1 at installation time). With this data, you can examine various DLFM system
1 configuration details, plus other DLFM-related data on a Data Links server. By
1 default, the output is stored in a file in the directory where the dlfm_dump utility
1 is invoked. The data that gets placed into the output file includes:

1 - The version of the current `DLFM_DB`.
1 - The keys currently in use for access-token generation.
1 - Security control information.
1 - Registered databases and prefixes.
1 - Data Links File System (DLFS) directory tree structures.
1 - DATALINK columns that reference this Data Links server.
1 - DB2 backups involving linked files on this Data Links server.
1 - Lists of all linked and unlinked files on this Data Links server.

1 **Important:**

1 - If the DLFM is managing a large number of files, both those that are currently
1 linked as well as those that were previously linked, the output file could be very
1 large. Ensure that you have sufficient space in the file system where the output
1 file is to be written. You can use the amount of space that the `DLFM_DB` occupies,
1 or that a backup of the `DLFM_DB` occupies, to estimate the space that is required
1 for the dlfm_dump output file.
1 - Because the output of the dlfm_dump utility contains sensitive security
1 information, ensure that you place the output file in a secure directory.

1 You use the following syntax to invoke the dlfm_dump utility.

1 **Requirement:** You must execute this command from the Data Links server using
1 the Data Links Manager Administrator user ID.

```
►►──dlfm_dump── -all──┬──────────────────────┬──────────────────────────────►◄
                      └─ -o──output_filename─┘
```

-all A required parameter, which specifies that all significant DLFM definitions and data are to be placed into an output file.

-o *output_filename*
An optional parameter, which specifies the name of the output file where the data will be saved. The default output file name is `dlfm_dump.log`.

The dlfm_dump utility output does not include temporary data that the `DLFM_DB` might be maintaining, such as data that is stored during transaction processing.

Note: The dlfm_dump utility output is minimally formatted, and is primarily intended to assist in diagnosing problems, not for general reporting purposes. To produce more readable output of Data Links server data, use any or all of the following commands:

- **dlfm retrieve**
- **dlfm list registered databases**
- **dlfm list registered directories**
- **dlfm list registered prefixes**
- **dlfm list registered replication access control**
- **dlfm list registered users**
- **dlfm list upd_in_progress files for db**
- **dlfm list upd_in_progress files for prefix**

DB2 Data Links logging manager

When you report problems to IBM® Service, you might be asked to perform a DB2® trace to provide the service personnel with details of your DB2 Data Links Manager environment. DB2 traces can be especially useful for analyzing a recurring and reproducible problem.

The Data Links server contains its own DB2 database, which is used as a logging manager to track all linked files. Therefore, you can take a DB2 trace on both the DB2 host and Data Links server machines where a problem was encountered, if necessary. In certain situations, you might need to run a *concurrent* DB2 trace on several machines. For example, you might encounter a communication problem between a DB2 host and a Data Links server machine.

During the process of performing a DB2 trace, all actions and any relevant parameter values get logged. Traces should be taken when there is minimum activity on the machine to prevent the capture of unnecessary information.

The process of performing a trace has a global effect on the behavior of a DB2 instance. The degree of performance degradation is dependent on the type of problem and on how many resources are being used to gather the trace information.

DB2 Data Links server failure

In the case of a machine failure on a Data Links server, DB2 applications interacting with the Data Links File Manager can hang. Use the **db2 force application** command to force DB2 applications off the system.

Also, the following components might require recovery after a Data Links server machine fails:

- The DB2 database containing a table which has a DATALINK column
- The Data Links Manager database (DLFM_DB)
- Data Links Filesystem Filter (DLFF)
- File systems under the control of the DLFF that are registered to the Data Links Manager
- The Data Links Manager archive area

DB2 Data Links failure and recovery overview

If a disk that contains files referenced by a DATALINK column fails, all the user files, along with the directory hierarchy of the file system, might be destroyed. To recover from such a scenario, the administrator should make periodic backups of the file system containing the user data and directory hierarchy, so that it can be restored. The restored file system must preserve directory and file ownerships, and time stamps.

After restoring the file system, the directory structure must be brought up to the point-in-time of the crash by applying the directory changes that occurred after the file system backup was taken. After this step, the DB2® **RECONCILE** command must be run on all tables referencing files on the damaged disk. The *db2_recon_aid* utility is provided to simplify this task.

Following a crash, there are three possible file states:

1. Files that are in linked state, and are referenced in a DATALINK column with the RECOVERY NO attribute, are treated as follows:
   - If the file is not found on the file system, the corresponding `DATALINK` column value will be set to `NULL.`
   - If the file is found, and the DATALINK column also has the `READ PERMISSION FS` and `WRITE PERMISSION FS` attributes defined, no additional checks will be made to validate the correctness of the file.
   - If the file is found, and the referencing DATALINK column also has the `WRITE PERMISSION BLOCKED` attribute defined, the file's modification time and size will be checked. If there is a mismatch in the values, the `DATALINK` column value will be set to `NULL.`
2. Files that are in a linked state, when the corresponding DATALINK columns have the `RECOVERY YES` attribute, will be restored from the archive server if the file modification time is different than the file modification time at link time, or if the file is not found.

   If the modification time of the version on the file system is different, it is renamed with extension .MOD so that the more recent changes are not lost. The archived version is still retrieved, and the renamed version is reported in the exception report.

   If a renamed version of the file with .MOD extension already exists, the file will not be retrieved, the DATALINK column value will be changed to `NULL`, and it will be reported in the exception report.
3. Files that are in the unlinked state on the file server are not restored or checked for correctness.

Determining the file system directories that are needed to restore to the current point in time

After a file system restore operation, you must manually bring the file system directory hierarchy to the current point in time by recreating directories as necessary.

Use the Data Links Manager `fsysadm.log` file to help you determine directories that you need to recreate. Data always gets appended to the `fsysadm.log`.

On AIX® and Solaris™ Operating Environments, the directory changes are logged in the *`INSTHOME`*`/sqllib/fsysadm.log` file, where *INSTHOME* is the home directory of the Data Links Manager Administrator. There is one entry for each event. Setting the attributes of a file is also logged. The format of the entries for the `fsysadm.log` file is as follows.

```
Time = <timestamp> EUID = <integer> UID = <integer> GID = <integer> Mode = <octal>
 Action = <CREATE/REMOVE/SETATTR/RENAME> Object type = <DIR/FILE> Path = <fully qualified
source name, destination name>
```

where:

- *Time* is the time of the activity in local time
- *EUID* is the effective user ID of the user performing the action
- *UID* is the user ID attribute of the file or directory that was created, or whose attributes were modified
- *GID* is the group ID attribute of the file or directory that was created, or whose attributes were modified
- *Mode* is the octal representation of the mode of the file or directory

where Action can be:

- *CREATE* indicates a directory was created
- *REMOVE* indicates the file or directory was removed
- *SETATTR* indicates the mode of the file or directory was modified by the user
- *RENAME* indicates the file was renamed

where Object type can be:

- *DIR* the directory
- *FILE* the file

and where Path is the fully qualified path of the file or directory. If the action was RENAME, the destination name is displayed after the path information.

On Windows® systems, the directory changes are logged in the *`x:`*`\sqllib\dlfm\fsysadm.log` file, where *x:* represents the drive where you installed DB2® Data Links Manager. A single event can have multiple entries, depending upon how many users or groups of users have Access Control Lists for the given file or directory. The format of the first entry for the `fsysadm.log` file is as follows.

```
Time = <timestamp> User = <string> Action = <CREATE/REMOVE/SETATTR/RENAME>
  Object type = <DIR/FILE> Path = <fully qualified source name, destination name>
```

The format of any additional entries associated with the first entry is as follows:

```
ACE User = <string> Access = <Hex integer>  ACE Type = <Hex integer>
   ACE Flags =<Hex integer>
```

where:

- *Time* is the time of the activity in local time
- *User* is the name of the user performing the action
- *Owner* is the name of the owner of the file or directory
- *Path* is the fully qualified path of the file or directory
- *ACE User* is the name of a user who has an ACL entry for this file or directory
- *Access** is the set of flags indicating the types of access the user has
- *ACE Type** is the type of ACE (for example, allow/deny)
- *ACE Flags** is a set of ACE type-specific control flags

where Action can be:

- *CREATE* indicates a directory was created
- *REMOVE* indicates the file or directory was removed
- *SETATTR* indicates the mode of the file or directory was modified by the user
- *RENAME* indicates the file was renamed

where Object type can be:

- *DIR* the directory
- *FILE* the file

and where Path is the fully qualified path of the file or directory. If the action was RENAME, the destination name is displayed after the path information.

(*) For the definitions of these hexadecimal values, refer to the Access Control Entry structures in the Microsoft® SDK documentation for Windows NT® and Windows 2000.

Reconciling database tables with DB2 Data Links file date using the db2_recon_aid utility

The `db2_recon_aid` utility provides a mechanism for checking and running `RECONCILE` on tables of a database that are potentially inconsistent with the `DATALINK` file data on the file server.

Like the RECONCILE utility, the db2_recon_aid utility must be run on a DB2 server containing tables with DATALINK columns to be reconciled.

On AIX systems or Solaris Operating Environments, the `db2_recon_aid` utility is located in the *`INSTHOME`*`/sqllib/adm` directory, where *INSTHOME* is the home directory of the instance owner.

On Windows systems, the `db2_recon_aid` utility is located in *`x:`*`\sqllib\bin` directory, where *x:* is the drive where you installed DB2 Data Links Manager.

To run the `db2_recon_aid` utility, use the following syntax:

```
▶▶—db2_recon_aid————————————————————————————————————————————————————▶
```

```
   ┌─────────────────────────────────────────────────────────────────────────────┐
   ▼                                                                             │
►►──┬─ -db──database_name─────────────────────────────────────────────────────┬─┴──────────►◄
    ├─ -check─────────────────────────────────────────────────────────────────┤
    ├─ -reportdir──directory_name─────────────────────────────────────────────┤
    └─ -selective── -server──data_links_server_name── -prefixes──prefix_list──┘
```

database_name

A required value, which specifies the name of the database containing the tables with DATALINK columns that need to be reconciled.

-check A parameter that instructs the utility to list the tables that might need reconciliation. If you use this parameter, no reconcile operations will be performed. This parameter is required when the **-reportdir** parameter is not specified.

-reportdir *directory_name*

Required when the **-check** parameter is not specified. Specifies the directory where the utility is to place a report for each of the reconcile operations. For each table on which reconcile was performed, files of the format <tbschema>.<tbname>.<ext> will be created where:

- <*tbschema*> is the schema of the table.
- <*tbname*> is the table name.
- <*ext*> is `.ulk` or `.exp`. The `.ulk` file contains a list of files that were unlinked on the Data Links server, and the `.exp` file contains a list of files that were in exception on the Data Links server.

-selective

An optional parameter that instructs the utility to process only those tables with DATALINK columns containing file references that match the specified **-server** and **-prefixes** criteria.

- If you use this parameter, you must also both the -server and -prefixes parameters.
- If you do not use this parameter, then all Data Links servers and their prefixes that are registered with the specified DB2 database will either be reconciled, or will be flagged as needing reconciliation.

-server *data_links_server_name*

Required when the **-selective** parameter is used. Specifies the name of the Data Links server for which the reconcile operation is to be performed. The name value must be an IP hostname that is identical to the Data Links server hostname registered with the specified DB2 database.

If this parameter is not used, all Data Links servers that are registered with the specified DB2 database will be reconciled.

-prefixes *prefix_list*

Required when the **-selective** parameter is used. Specifies the name of one or more Data Links File System (DLFS) prefixes. Prefix values must start with a slash, and must be registered with the specified Data Links file server. Separate multiple prefix names with a colon (:), but do not include any embedded spaces. For example: `/dlfsdir1/smith/:/dlfsdir2/smith/`.

The path in a DATALINK column value is considered to match the *prefix_list* if any of the prefixes in the list are a left-most substring of the path.

If this parameter is not used, all prefixes for all Data Links servers that are registered with the specified DB2 database will be reconciled.

Examples:`db2_recon_aid -db STAFF -check`

```
db2_recon_aid -db STAFF -reportdir /home/smith
```

```
db2_recon_aid -db STAFF -check -selective -server
dlmserver.services.com  -prefixes /dlfsdir1/smith/
```

```
db2_recon_aid -db STAFF -reportdir /home/smith -selective -server
dlmserver.services.com -prefixes /dlfsdir1/smith/:/dlfsdir2/smith/
```

DB2 Data Links Manager recovery scenarios

This topic presents some sample DB2® Data Links Manager failure scenarios and the steps required to recover from them.

The scenarios use the following terms:

DLFS file system
Registered prefix (example for AIX®: /dlink)

DLFM backup directory
Directory where files are backed up (example: `/dlfm/dlfm_backup`)

DLFM instance directory
Instance directory of the Data Links Manager Administrator ID (example: `/home/dlfm`)

DLFM DB2 database
DB2 database that contains all metadata (DLFM_DB)

DB2 database
Registered database that contains DATALINK data type (example: CROWN)

The example DB2 database is referred to as ″CROWN″ throughout all scenarios.

Important: Some of the following scenarios might require operations to be performed on both the DB2 node and the DB2 File Manager node.

Scenario	Recovery Steps

DB2 database is lost or was accidentally dropped, but DB2 backup and log files are available.
Important: When a database gets dropped, it happens within the period of time as specified by the DB2 database configuration parameter *dl_time_drop*.
In this example, the DB2 database has **not** been dropped from the Data Links server.
	1. On the DB2 host, enter the following commands. As a result, all of the affected tables will be put into the Datalink_Reconcile_Not_Possible (DRNP) state.
	`db2 "restore database CROWN"`
	`db2 "rollforward database CROWN to end of logs and stop"`
	`db2 "connect to CROWN"`
	2. Place all tables with DATALINK columns into Datalink_Reconcile_Pending (DRP) state with the following commands:
	`db2 set integrity for <table> to datalink reconcile pending`
	`db2 set integrity for <table> datalink reconcile pending immediate unchecked`
	`db2 reconcile <table> dlreport <filename>`

Scenario	Recovery Steps

DB2 database was explicitly dropped, but DB2 backup and log files are available.
Important: When a database gets dropped, it happens within the period of time as specified by the DB2 database configuration parameter *dl_time_drop*.
	Requirement: You *must* take a backup of your DLFM_DB database before updating it as described in the following procedure.
	Recommendation: Work with IBM® Service in updating your DLFM_DB database. This database is an essential part of your Data Links Manager configuration.
	1. Ensure that the drop database operation is complete, and that all files associated with that database have been unlinked.
	2. On the Data Links server enter the following commands. It is critical that you set the `dbid` value in the `db2 update` statement to be exactly as found in the `db2 select` statement.
	`db2 "connect to dlfm_db"`
	`db2 "select dbid, dbname, dbinst, hostname from dfm_dbid"`
	`db2 "update dfm_dbid set action=5 where dbid=x'35B3D7BE0006BF7B'"`
	3. On the DB2 host, enter the following commands. As a result, all of the affected tables will be put into the Datalink_Reconcile_Not_Possible (DRNP) state.
	`db2 "restore database CROWN"`
	`db2 "rollforward database CROWN to end of logs and stop"`
	`db2 "connect to CROWN"`
	4. For each table placed in DRNP state in step 3, enter the following commands:
	`db2 set integrity for <table> to datalink reconcile pending`
	`db2 set integrity for <table> datalink reconcile pending immediate unchecked`
	`db2 reconcile <table> dlreport <filename>`

The DLFM_DB database is lost, but the backup and all log files for the DLFM_DB database are available.	1. On the Data Links server enter the following commands:
	`db2 "restore database dlfm_db"`
	`db2 "rollforward database dlfm_db to end of logs and stop"`
	2. On the DB2 host, enter the following command to run the **db2_recon_aid** utility. This utility automatically runs RECONCILE for each table with URL file references to the affected Data Links server:
	`db2_recon_aid -db CROWN -reportdir <dirpath>`
	` -selective -server <dlm_hostname> -prefixes <dlfs_prefix>`
	• **dlm_hostname** is the registered IP hostname of the affected Data Links Manager
	• **dlfs_prefix** is the registered prefix corresponding to the affected Data Links File System (DLFS)

The DLFM_DB database is lost, a backup of the DLFM_DB database is available, but not all of the log files are available.	1. On the Data Links server enter the following commands:
	`db2 "restore database dlfm_db"`
	`db2 "rollforward database dlfm_db to end of logs and stop"`
	2. On the DB2 host, enter the following commands. As a result, all of the affected tables will be put into the Datalink_Reconcile_Pending (DRP) state.
	`db2 "restore database CROWN"`
	`db2 "rollforward database CROWN to end of logs and stop"`
	`db2 "connect to CROWN"`
	3. Place all tables with data link values into DRP state by entering the following commands:
	`db2 set integrity for <table> to datalink reconcile pending`
	`db2 set integrity for <table> datalink reconcile pending immediate unchecked`
	`db2 reconcile <table> dlreport <filename>`

Scenario	Recovery Steps

The Data Links File System (DLFS) is lost.	1. Restore the DLFS from your storage manager.
	2. On the DB2 host, enter the following command to run the **db2_recon_aid** utility. Notice that by using the **-selective** options, you can perform reconciliation for just the DLFS that was lost.
	`db2_recon_aid -db CROWN -reportdir <dirpath>`
	` -selective -server <dlm_hostname> -prefixes <dlfs_prefix>`
	• **dlm_hostname** is the registered IP hostname of the affected Data Links Manager
	• **dlfs_prefix** is the registered prefix corresponding to the affected Data Links File System (DLFS)

The DLFM backup directory is lost.	Restore the DLFM backup directory from your storage manager.

The DLFS file system and the DLFM backup directory are lost.	1. Restore the DLFM backup directory from your storage manager.
	2. Restore the DLFS from your storage manager.
	3. On the DB2 host, enter the following command to run the **db2_recon_aid** utility. Notice that by using the **-selective** options, you can perform reconciliation for just the DLFS that was lost.
	`db2_recon_aid -db CROWN -reportdir <dirpath>`
	` -selective -server <dlm_hostname> -prefixes <dlfs_prefix>`
	• **dlm_hostname** is the registered IP hostname of the affected Data Links Manager
	• **dlfs_prefix** is the registered prefix corresponding to the affected Data Links File System (DLFS)

The DLFM_DB database, the DLFM backup directory, and the DLFS file system are lost, but the backup and all log files for the DLFM_DB database are available.	1. On the Data Links server enter the following commands:
	`db2 "restore database dlfm_db"`
	`db2 "rollforward database dlfm_db to end of logs and stop"`
	2. Restore the DLFM backup directory from your storage manager.
	3. Restore the DLFS from your storage manager.
	4. On the DB2 host, enter the following command to run the **db2_recon_aid** utility. Notice that by using the **-selective** options, you can perform reconciliation for just the DLFS that was lost.
	`db2_recon_aid -db CROWN -reportdir <dirpath>`
	` -selective -server <dlm_hostname> -prefixes <dlfs_prefix>`
	• **dlm_hostname** is the registered IP hostname of the affected Data Links Manager
	• **dlfs_prefix** is the registered prefix corresponding to the affected Data Links File System (DLFS)

Scenario	Recovery Steps

The DLFM_DB, the DLFS file system, and the DLFM backup directory are lost. The backup of the DLFM_DB database is available, but not all log files are available.	1. On the Data Links server enter the following commands:
	`db2 "restore database dlfm_db"`
	`db2 "rollforward database dlfm_db to end of logs and stop"`
	2. Restore the DLFM backup directory from your storage manager.
	3. Restore the DLFS from your storage manager.
	4. On the DB2 host, enter the following command to run the **db2_recon_aid** utility. Notice that by using the **-selective** options, you can perform reconciliation for just the DLFS that was lost.
	`db2_recon_aid -db CROWN -reportdir <dirpath>`
	` -selective -server <dlm_hostname> -prefixes <dlfs_prefix>`
	• **dlm_hostname** is the registered IP hostname of the affected Data Links Manager
	• **dlfs_prefix** is the registered prefix corresponding to the affected Data Links File System (DLFS)

The DB2 database, the DLFM_DB database, the DLFS file system and DLFM backup directory are lost, but backup, and all log files for the DLFM_DB database are available.	1. On the Data Links server enter the following commands:
	`db2 "restore database dlfm_db"`
	`db2 "rollforward database dlfm_db to end of logs and stop"`
	2. Restore the DLFM backup directory from your storage manager.
	3. Restore the DLFS from your storage manager.
	4. On the DB2 host, enter the following commands. As a result, all of the affected tables will be put into the Datalink_Reconcile_Not_Possible (DRNP) state.
	`db2 "restore database CROWN"`
	`db2 "rollforward database CROWN to end of logs and stop"`
	`db2 "connect to CROWN"`
	5. For each table placed in DRNP state in step 4, enter the following commands to place them in DRP state:
	`db2 set integrity for <table> to datalink reconcile pending`
	`db2 set integrity for <table> datalink reconcile pending immediate unchecked`
	`db2 reconcile <table> dlreport <filename>`
	Note: Rollforward to a point-in-time might not put tables that have all DATALINK columns defined with RECOVERY NO into Datalink_Reconcile_Pending (DRP) state. For all such tables, run the RECONCILE utility. You can also use the db2_recon_aid utility to automatically identify and run RECONCILE against those tables.

Business intelligence issues

DB2 Data Warehouse Center logging

The log holds records until a designated count limit is reached. When the count limit is reached, the Data Warehouse Center automatically deletes the oldest logs.

Recommendation: Set the log record count to a size that holds 3 to 4 days worth of records.

You cannot turn off the basic logging function.

Viewing DB2 Data Warehouse Center runtime errors

Use the basic logging function to view run-time (step processing) errors.

Procedure:

To view run-time errors:

1. Open the Data Warehouse Center desktop.
2. Click **Data Warehouse Center -> Work in Progress**.

   The Work in Progress window opens.
3. Select the step for which you want to view errors.
4. To open the Log Viewer window and display the run-time errors for the selected step, click **Log**.

Viewing DB2 Data Warehouse Center build-time errors

Use the basic logging function in the Data Warehouse Center to view build-time errors.

Procedure:

To view build-time (table import, object creation, and step promotion) errors:

1. Open the Work in Progress window.
2. Click **Work in Progress -> Show Log** to open the Log Viewer window and display the build-time errors for the Data Warehouse Center.

Viewing log entries in the Data Warehouse Center

If a step or process does not run successfully, you can use the Log Viewer to find the cause of the failure.

Procedure:

To view log entries:

1. Select the step or process.
2. Click **Log** to open the Log Viewer window.

DB2 Data Warehouse Center component trace data

Running warehouse agents as a user process (Windows)

If you run a warehouse agent as a system process instead of a user process, it can cause steps to fail. When the warehouse agent runs as a system process, it is not authorized to connect to network drivers or products because the process does not have a user ID. If the warehouse agent runs as a user process, the warehouse agent has the characteristics of a user, including the ability to access network drives or programs to which the user is authorized.

Procedure:

To run the warehouse agent as a user process:

1. Change the warehouse server, warehouse logger, and warehouse agent daemon services to run as user processes by performing the following steps:
   a. Double-click the **Services** icon in the Windows **Control Panel** folder.
   b. Stop the services.

c. Select the service, and click **Startup**.
d. Click **This Account**.
e. Click the **...** button after the **This Account** field to select a user ID.

The user ID must have administration authority in Windows NT, Windows 2000, or Windows XP and authorization to any required network drive.

f. Type the password for the user ID in the appropriate fields.
g. Click **OK**.
h. Restart the services.

2. If you are using a supplied OLAP server program, verify that the DB2 OLAP or Essbase client is installed on a drive that is local to the agent that is running the program.
3. If you are using a supplied OLAP server program, verify that the ARBORPATH variable (set on the Essbase client or administrator) specifies a drive that is local to the agent that is running the program and is specified as a system variable.

Running a Data Warehouse Center component trace

Follow this procedure to run a Data Warehouse Center component trace.

Procedure:

To run a Data Warehouse Center component trace:

1. Right-click the warehouse object, and click **Properties**.
2. Specify the trace level for the warehouse control database, ODBC connection, server, agent, or logger as directed by IBM Software Support.
3. Click **OK**.
4. Restart the services as requested.
5. Perform the failing operation.
6. Repeat steps 1 through 4 to set the trace level back to 0.

After completing this procedure, turn the trace level back to 0 to prevent performance degradation.

You can run an agent trace independently for individual steps by setting the trace level in the step's Properties notebook on the Processing options page.

Error logging for warehouse programs and transformers

The supplied warehouse programs and transformers write errors to log files.

Warehouse programs

Supplied warehouse programs write data to the directory that is specified in the VWS_LOGGING environment variable. Clear the directory of the log files after sending the log files to IBM® Software Support.

Transformers

Transformer error messages start with DWC14. Transformer error messages, warning messages, and returned SQL codes are stored as secondary codes. For example, if a message starts with DWC14, a transformer (stored procedure) caused the error. If the secondary code includes an SQLCODE, an SQL statement in the transformer caused the error. To enable logging, specify a log table name on the Processing

Options page of the Properties notebook for the step, and add a suffix of *:n* to the log table name. The value of *n* indicates the logging level:

0 No logging

1 Log errors only

2 Log errors and warnings (this is the default logging level)

3 Log errors, warnings, and informational messages (for example, starting and stopping a transformer)

For example, to indicate a log table named MyLogTable that contains log entries at log level 3 or less, specify `MyLogTable:3`. In the output log tables, the message type is one of the following values:

E Error

W Warning

Q SQL code

You can include a table space name after the log table name by appending the log level to the table space name.

For example, to indicate a log table named MyLogTable that is located in the MyTableSpace table space and contains entries at log level 3 or less, specify `MyLogTable,MyTableSpace:3`.

The output log table in the warehouse control database contains detailed error messages, warning messages, and SQL codes. In the output log tables, the message type is one of the following values:

E Error

W Warning

Q SQL code

Tracing errors created by the Apply program

You can trace errors that are created when you use the Apply program.

Procedure:

To enable tracing for the Apply program, set the Agent Trace value = 4 in the Warehouse Properties page. The Agent turns on full tracing for Apply when Agent Trace = 4.

If you do not see any data in the CD table, then the Capture program is not started, or you did not create changed data by updating the source table.

Start error trace files

The Data Warehouse Center creates three log files automatically when the logger is not running. The log file names are IWH2LOGC.LOG, IWH2LOG.LOG, and IWH2SERV.LOG. The Data Warehouse Center stores the files in the directory that is specified by the VWS_LOGGING environment variable.

The log files are:

IWH2LOGC.LOG
When the logger is not running, processes write messages to this file. The

Data Warehouse Center server and the OLE server write to this file. The file is created only if the logger stops. The file contains the complete content of all messages that could not be sent.

IWH2LOG.LOG
The logger creates this file when it cannot start itself or when a trace is activated. Key diagnostic information is written to this file when the logger cannot start itself, and cannot write to the Data Warehouse Center log. If you hear five beeps, or receive an application error, when the logger stops, look in this file. The server cannot start if the logger cannot start.

IWH2SERV.LOG
The server log contains the startup message and grows when the server trace is on.

Information Catalog Center issues

Finding help for information catalog center issues

The Information Catalog Center gives you some resources to help you solve problems. These resources are:

- Online information and messages.

  The Information Catalog Center provides extensive online information and messages to help you solve problems. When you or your users receive a message, use the online help first to resolve the problem.

  You can find help for Information Catalog Center messages and explanations in the *Message Reference*.

  You can also look up message help from a DB2 command line by typing `db2 Information Catalog Center ICM`**`nnnn`**where **nnnn** represents the id number of the message.
- Information Catalog Center trace file.

Recovering Information Catalog Center components and data

If you experience a hardware or software failure, you can lose your information catalog database, your descriptive data, and parts of the component. If you backed up the necessary components and data, you can restore your system, the Information Catalog Center, and data.

Steps:

If a system failure occurs, perform the following steps after the database server's hard disk is restored and before your users access the information catalog:

1. Recover your database management system and reinstall the Information Catalog Center, as necessary.
2. Restore the information catalog databases by using your backup files.

DB2 Cube Views issues

Database connection behavior using the OLAP Center

If you cannot connect to a database using the OLAP Center, check that the version of DB2 Cube Views matches the version of the metadata tables in the DB2 catalog.

The following table shows how the OLAP Center behaves when the versions of DB2 Cube Views and the metadata tables in the DB2 catalog are mismatched.

Version of DB2 Cube Views	Version of the metadata tables in the DB2 catalog	Behavior of the OLAP Center
Not installed	Not applicable	Connection fails and the OLAP Center displays an error message
Version 8.1	None	Connection fails and the OLAP Center displays an error message
Version 8.1	Version 8.1	Connection fails and the OLAP Center displays an error message
Version 8.2	None	The OLAP Center can configure the database for use with DB2 Cube Views, Version 8.2
Version 8.2	Version 8.1	The OLAP Center can migrate the database for use with DB2 Cube Views, Version 8.2
Version 8.2	Version 8.2	Connection is successful.

DB2 Cube Views backward compatibility with the OLAP Center and metadata API

DB2 Cube Views Version 8.2 has limited backward compatibility support for OLAP Center and the metadata API.

DB2 Cube Views API supports the following:

- Version 8.2 API supports Describe requests from a Version 8.1 client. Describe is the only Version 8.1 operation that is supported from the previous release.

DB2 Cube Views OLAP Center supports the following:

- Version 8.2 OLAP Center does not support Version 8.1 API or Version 8.1 metadata tables.
- Version 8.1 OLAP Center does not support Version 8.2 API or Version 8.2 metadata tables.
- Version 8.1 OLAP Center cannot connect to a Version 8.2 DB2 database.
- OLAP Center supports the import of Version 8.1 XML. When you import a Version 8.1 XML file, OLAP Center migrates the XML to Version 8.2, using the Translate operation, before importing the metadata.
- OLAP Center can export both Version 8.2 and Version 8.1 XML.

DB2 Cube Views db2mdapiclient supports the following:

- Version 8.1 db2mdapiclient supports Version 8.2 API if you use Version 8.2 XML.
- Version 8.2 db2mdapiclient supports Version 8.1 API if you use Version 8.1 XML.

Improving queries issued to federated data sources

If your queries issued to a remote data source do not improve as you expected after you optimize, ensure that the federated system is set up correctly for DB2 Cube Views.

You must complete the following steps before you can expect queries issued to remote data sources to improve:

- Enable your federated system for DB2 Cube Views.
- Create a complete cube model that satisfies the base rules, cube model completeness rules, and optimization rules described in Metadata object rules.
- Optimize a cube model.
- If query performance does not improve, check the steps described in Troubleshooting summary tables.

If, after completing the steps above, the performance of your queries still do not improve, consider the following issues:

- Make sure that all applicable constraints are defined.
- Make sure that the DB2_MAXIMAL_PUSHDOWN setting is set to yes as described in Defining remote data sources.
- Consider collocating the dimension tables that are involved in the queries on your federated server. Collocating the dimensions, so that a replicated copy of the dimension table exists on the federated server, might improve performance.

Connectivity issues

Before beginning to analyze a connectivity problem, it is imperative that you understand your environment. You should know three pieces of information (as a minimum):

1. Operating systems
2. DB2 products, versions and FixPak levels
3. Communication protocols used between each tier

If you are working in a multi- tier model (i.e. the client tier is supplemented by one or connectivity servers) then it often helps to draw a diagram of your environment. For an example of a connectivity server diagram, see DB2 Connect Enterprise Edition as a connectivity server.

Two terms which you will come across when investigating connectivity issues are: DRDA Application Requestor (AR) and DB2 UDB DRDA Application Server (AS). These terms come from DRDA (Distributed Relational Database Architecture), which is a set of protocols used by DB2 to coordinate communication. The application requester is the code that handles the application side of a distributed connection; it is the machine that you are connecting ″from″. The application server is the code that handles the database side of the connection; it is the machine that you are connecting ″to″.

If you are connecting to a host system, the term ″database″ may be unclear, since that term is used differently on some mainframes. Refer to Host databases for clarification of the term.

Click on this link for more information on Supported and non-supported client configurations.

Click on this link for more information on Client-to-Server communication scenarios.

Problem determination process for connectivity

Connectivity problems often involve multiple software, hardware and communications products. Problem determination is best approached by a process of elimination and refinement of the available data to arrive at a conclusion (the location of the error).

Problem determination includes narrowing the scope of the problem and investigating the possible causes. The proper starting point is to gather the relevant information and determine what you know, what data has not been gathered, and what paths you can eliminate. At a minimum answer the following questions.

- Are the communication paths operational?
- Has the initial connection been successful?
- Is the problem intermittent or persistent?
- Have there been any communication network changes that would make previous directory entries invalid?
- Has the database been started?
- Is the communication breakdown between client and DB2 Connect workstation, DB2 Connect workstation and host or iSeries(TM) database server, all clients or one client?
- Is the problem only encountered within a specific application?
- What can you determine by the content of the message and the tokens returned in the message?
- Will using diagnostic tools provide any assistance at this time?
- Are other machines performing similar tasks working correctly?
- If this is a remote task, is it successful if performed locally?

Some of these questions are particularly important, and will be expanded on.

Determining if communication paths are operational

It is important to isolate whether the problem you are encountering is limited to DB2. For example, if you are encountering network errors, is it only via DB2 connections that you are encountering network errors? This becomes especially important when the error is intermittent, as there is a greater likelihood that the problem is outside of DB2 in those situations.

There are tools specific to each protocol that can be used to verify that communication between the participating machines is possible external to DB2. Here are some examples:

TCP/IP:

```
ping <hostname or ip address>
```

Try it out! Your successful result should look something like this:

```
Pinging myserver [1.23.45.678] with 32 bytes of data:
Reply from 1.23.45.678: bytes=32 time=921ms TTL=253
Reply from 1.23.45.678: bytes=32 time=721ms TTL=253
Reply from 1.23.45.678: bytes=32 time=942ms TTL=253
Reply from 1.23.45.678: bytes=32 time=80ms TTL=253

Ping statistics for 1.23.45.678:
 Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milliseconds:
 Minimum = 80ms, Maximum = 942ms, Average = 666ms
```

APPC:

Each different APPC communication product will have its own method of verifying that a successful LU to LU session can be established. For example, on AIX, you could start the SNA subsystem, start the node, start the link station, and then test a session, all via the smit utility. Please refer to your specific APPC communication product's documentation for more details.

Named Pipes:

Try issuing a net use command from the client to a shared folder on the server.

The PCT tool

To assist in debugging the communication setup between the client and the server, DB2 provides a tool called the Protocol Communications Tester. The tool is launched by means of the pct command, which is found in the sqllib/bin directory. It is a standalone tool designed to verify that a given LAN protocol is functional, and that two endpoints can communicate with each other via that LAN protocol.

The tool can help solve protocol configuration-type problems because it effectively removes DB2 from the picture until communication has been proven successful. If you cannot pass these communication tests that are external to DB2, this indicates that either your error is in the network layer or else you have not used correct protocol-specific values in the tool. DB2 will not be able to connect across this network until the network problems are corrected and the proper protocolspecific values are identified.

For usage information related to this tool, please refer to the readme.pct file located in the same directory as the executable file. The PCT tool can test the following protocols:

- NetBIOS
- TCP/IP

Verifying the initial connection

If it is a first-time setup, there is a strong likelihood that this is a configuration issue. First-time connection problems caused by configuration errors are most often also persistent. To resolve the issue, verify that the setup has been properly performed. If the connection worked previously, then the questions you should ask yourself are:

- How long ago did it work?
- What has changed?

These types of questions will help you to pinpoint what network changes or fix pack level upgrades, application changes or system outages may have taken place between the point in time when it last worked, and now. Identifying what changed and where (i.e. on the client, gateway, or server) helps you to determine where you should focus your energy during investigation.

Determining intermittent and persistent problems

If the problem is persistent, i.e. it can be consistently recreated with a set of simple steps, and then it is most likely a configuration issue. Your focus should be on confirming the setup on the client and server. Refer to Confirming connection configuration.

If the problem is intermittent, then it will be a bit more difficult to debug. It is not the error message itself that is different -- usually you will get an SQL30081 error message just like you do for persistent problems. The difficulty lies in the fact that you cannot recreate it at will, and thus have fewer chances of capturing and diagnosing the error. To pursue intermittent error messages, you will need to investigate the circumstances of the error, for example:

- What is the network state at the time of the problem?
- What is the frequency of the occurrence?
- Is there a patterns as to what events are occurring at the time of the problem?

Depending on how frequently the errors are occurring, traces may be hard to obtain. It is understandable that you may not want to have traces running for extended periods of time just to capture the error -- particularly if the problem is occurring on a production system. Thus you frequently need to rely on the information in the db2diag.log file for these types of problems. Intermittent errors, idle threads, connection pooling and connection concentrator describes additional items to take into consideration when investigating intermittent connectivity errors.

Determining the location of the communication breakdown

A simple way to begin to answer this question is to determine whether or not you can connect directly on the server. This is a very key question, and should be something that you ask yourself every time that you encounter a communication error. If you cannot connect locally at the time of the error, then you can eliminate the client and the communication layer entirely from the problem scenario and can focus instead on recreating and diagnosing the error directly on the server.

If a local connection is indeed successful, then the next step is to simulate a remote connection directly on the server. This is done to reduce the layers of complexity by recreating the entire problem on one machine. This is often referred to as a loopback. The loopback is just a method of cataloging the local database as if it were on a remote machine, as described here. Example: Creating a loopback connection

If the connection works via this loopback connection test, but fails from a remote client, then you might want to investigate these areas:

1. Is there a firewall between the client and the server?
2. Are the node and database catalogs on the client similar to those used in the loopback node. Note that the service name (if used) could be different in the client's node catalog, but that the port number that it maps to on each machine should nonetheless match.

If the connection fails via the local loopback test, then you can ignore the client entirely from this point on. If the error is persistent, it is likely a configuration issue, and the server is not correctly set up to listen for incoming requests. You would thus need to focus on confirming the server configuration values. Refer to Confirming connection configuration.

If you can only recreate the problem from remote clients, then you should determine whether all clients or one client experiences the problem.

- If the problem occurs only for a subset of the clients connecting to this server, then you should identify the commonalties and differences between these clients, and compare their configurations (DB2 and network).
- If the problem is occurring across all of your clients, then a simple comparison is not an option for you.

- If the problem is intermittent but is happening across all of the servers at the same time, then this points to a greater likelihood that the error originates on the server or network, rather than the client.

If you have only encountered the problem within a single application, it is beneficial to try connecting from the DB2 command prompt. Whether or not this will help you narrow the scope of the problem will depend on whether or not the error is easily reproducible in the application. If you can consistently recreate the problem from your application, but can connect successfully from the DB2 prompt, then it would be wise to examine your application.

As you determine the answers to these questions, you should find that you are successfully narrowing the problem description and thus increase the chances of determining where the root cause of the problem lies.

Testing connectivity using the PCT tool - example

This example illustrates the use of the TCP/IP version of the PCT tool. To test connections made over NETBIOS protocols, the PCT instructions are very similar. The main difference is that you use the command pctn instead of pctt.

All of the pct files can be found in `INSTHOME/sqllib/bin`, where `INSTHOME` is the home directory of the instance owner. On Windows systems, the files are located in the `sqllib\bin` subdirectory.

To perform the connectivity test, you must:

1. Set up PCT on the server
2. Start the PCT listener on the server
3. Set up PCT on the client
4. Launch a PCT communication test on the client.

Configuring PCT on the server

On your server, do the following:

1. Verify your server's hostname and IP address via the following command:

   ```
   pctt -r

   D:\sqllib\bin>pctt -r
   TCPIP Resources...
   +-------------------------------------------+
   | HostName : myserver                       |
   | HostAddress: myserver.ibm.com             |
   | IPAddress : 123.456.7.890                 |
   +-------------------------------------------+
   ```

2. Generate a pct.ini file: `pctt /gt` The resultant pct.ini file will appear as follows:

   ```
   #--------------------------------------------------------
   # PCT.INI
   # Protocol Communication Tester INI File
   # This file contains the Protocol Specific Configuration
   # Parameters & values to be used in conjunction with
   # the Protocol Communication Tester (PCTx).
   #
   # Defaults will be used if the parameter is not listed.
   #--------------------------------------------------------
   [TCPIP]
   ServerHostName =
   ```

```
ServerHostAddress =
ServerIPAddress =
ServiceName =
ServerPort = 49433
```

Ensure that you run this command in a directory to which you have write access, or else the pct.ini file will not be created successfully.

3. Edit the pct.ini file to reflect the protocol-specific parameter values for this server and for the particular instance that your client will be trying to connect to. When you use this tool to debug problems in DB2, it is important to configure the pct.ini file with the exact values that your DB2 instance will use to listen for incoming client requests, if possible. For example, you should set the `ServiceName` or `ServerPort` value in pct.ini to the same value as is found in the `SVCENAME` field in the output from the command.

```
"db2 get database manager configuration"
```

Note:

If you have set up PCT to use the same values as the DB2 instance uses, you must stop the instance while testing PCT, since PCT will return an error in situations where there is contention for ports that are already in use.

You only need to enter a value for either the ServerHostName or the ServerIPAddress. The same holds true for the ServiceName, ServerPort pair. Either one alone will suffice.

Starting the PCT listener on the server

The PCT tool must be running on the server prior to launching it on the client, otherwise the PCT tool on the client will fail, indicating that the connection was refused.

To make the PCT tool start listening for incoming requests, issue the command:

```
pctt s
```

When PCT starts successfully on the server, you see output like this:

```
- Reading configuration parameters
==> pct.ini file was found... using file specified protocol values!
 Protocol Tester values...
   Client/Server          : Server
   Connections            : 1
   Buffer Size            : 500
   Transaction Iterations : 1
   Trace                  : OFF
   Service                : Send/Recv/Verify
   Keep Connections       : NO
   Delayed Send (secs)    : 0
 Local TCPIP specific values...
          Hostname        : myserver
          HostAddress     : myserver.ibm.com
          IPAddress       : 123.456.7.890

- Initializing the Protocol            Date: 2-9-2005  Time: 22:54:52:97
| retcode = <    0>  ----[ TCPIP.socket     ]-----[  SUCCESS  ]------------
| retcode = <    0>  ----[ TCPIP.setsockopt ]-----[  SUCCESS  ]------------
| retcode = <    0>  ----[ TCPIP.bind       ]-----[  SUCCESS  ]------------
| retcode = <    0>  ----[ TCPIP.listen     ]-----[  SUCCESS  ]------------
- Listening for Remote Clients         Date: 2-9-2005  Time: 22:54:52:97
```

At this point the server side of the PCT tool is waiting to receive a communication from the client.

Configuring PCT on the client

On the client, do the following:

1. Generate a pct.ini file: `pctt /gt`. The file will be blank, and appear exactly as it did when performing this step on the server.
2. Edit this pct.ini file to reflect the protocol-specific parameter values for the server to which you want to connect. In order to use this tool to debug problems in DB2, it is important to fill out the pct.ini file with the exact values that you used when cataloging the remote DB2 server from the client. For example, if you used the server's hostname (as opposed to the IP address) when you ran the `CATALOG TCPIP NODE` command on the client, then do the same in the pct.ini file. Likewise if you used a servicename (as opposed to a port number), then do the same in pct.ini. Here is an example of an updated pct.ini file:

```
#---------------------------------------------------------
# PCT.INI
# Protocol Communication Tester INI File
# This file contains the Protocol Specific Configuration
# Parameters & values to be used in conjunction with
# the Protocol Communication Tester (PCTx).
#
# Defaults will be used if the parameter is not listed.
#---------------------------------------------------------
[TCPIP]
ServerHostName = myserver.ibm.com
ServerHostAddress =
ServerIPAddress =
ServiceName =
ServerPort = 50000
```

Starting a PCT communication test on the client

Issue the client connection attempt using the following command: `pctt c`

By default, PCT will perform an endpoint--to--endpoint connect operation, followed by send and receive operations with data verification, and finally it will disconnect.

Successful results appear as follows:

```
- Reading configuration parameters
==> pct.ini file was found... using file specified protocol values!
 Protocol Tester values...
   Client/Server         : Client
   Connections           : 1
   Buffer Size           : 500
   Transaction Iterations : 1
   Trace                 : OFF
   Service               : Send/Recv/Verify
   Keep Connections      : NO
   Delayed Send (secs)   : 0
 Local TCPIP specific values...
          Hostname        : myclient
          HostAddress     : myclient.ibm.com
          IPAddress       : 99.99.99.99
```

[(The results of a failed connection attempt are shown later.) The information in the above output shows the PCT tool's default settings as well as the protocol-specific information about your client machine.]

```
 Server TCPIP specific Values...
   Server Hostname       : myserver.ibm.com
   Server HostAddress    : myserver.ibm.com
   Server IPAddress      : 123.456.7.890
   Service Name          :
   Server Port           : 50000
```

[The information above was picked up from your client's pct.ini file and shows the values that will be used to identify the server.]

```
- Initializing the Protocol             Date: 02/09/05  Time: 19:59:17: 78
- Connecting to Remote System           Date: 02/09/05  Time: 19:59:17: 93
| retcode = <    0>  ----[ TCPIP.socket     ]-----[  SUCCESS  ]------------
| retcode = <    0>  ----[ TCPIP.connect    ]-----[  SUCCESS  ]------------
- Connection established!     1         Date: 02/09/05  Time: 19:59:19: 15
```

At this point, the connection step of the test has been completed.

```
- Sending Service Request               Date: 02/09/05  Time: 19:59:19: 15
| retcode = <    0>  ----[ TCPIP.send       ]---[ Bytes=  316 ]------------
- Receiving                             Date: 02/09/05  Time: 19:59:19: 15
| retcode = <    0>  ----[ TCPIP.receive    ]---[ Bytes=    4 ]------------
| retcode = <    0>  ----[ TCPIP.receive    ]---[ Bytes=  316 ]------------
- Server info:
    Computer Name   :
    Operating System:
    Version         :
    Hostname        :
- Send/Receive/Verify Data Loop         Date: 02/09/05  Time: 19:59:19: 78
- Sending Data                          Date: 02/09/05  Time: 19:59:19: 78
| retcode = <    0>  ----[ TCPIP.send       ]---[ Bytes=  500 ]------------
- Receiving                             Date: 02/09/05  Time: 19:59:19: 78
| retcode = <    0>  ----[ TCPIP.receive    ]---[ Bytes=    4 ]------------
| retcode = <    0>  ----[ TCPIP.receive    ]---[ Bytes=  496 ]------------
```

[By default, the PCT tool passes some data from the server to the client, to verify the connection. Then the connection is closed (see below).]

```
- Terminating the Connection            Date: 02/09/05  Time: 19:59:19:156
| retcode = <    0>  ----[ TCPIP.sock_close ]-----[  SUCCESS  ]------------
+-------------------------------------------------+
|  Protocol Connection Test Completed - SUCCESS  |
+-------------------------------------------------+
```

An example of an error message which can occur if PCT fails to complete the test successfully is as follows:

```
- Reading configuration parameters
==> pct.ini file was found... using file specified protocol values!
Protocol Tester values...
Client/Server : Client
Connections : 1
Buffer Size : 500
Transaction Iterations : 1
Trace : OFF

Service : Send/Recv/Verify
Keep Connections : NO
Delayed Send (secs) : 0
Local TCPIP specific values...
Hostname : myclient
HostAddress : myclient.ibm.com
IPAddress : 99.999.9.999
Server TCPIP specific Values...
Server Hostname : myserver
Server HostAddress : myserver.ibm.com
Server IPAddress : 1.23.45.678
Service Name :
Server Port : 44000
```

[Note the incorrect port number used in this test.]

```
- Initializing the Protocol             Date: 02/09/05  Time: 20:07:23:406
- Connecting to Remote System           Date: 02/09/05  Time: 20:07:23:421
| retcode = <    0>  ----[ TCPIP.socket     ]-----[  SUCCESS  ]------------
| retcode = <10061> +----[ TCPIP.connect    ]-----[   ERROR   ]------------
+===== ERROR ========================================================
| retcode = <10061> ->  Connection refused
+====================================================================
```

The return code given here is a TCP/IP return code 10061 which maps to CONNECTION REFUSED. The cause of the error in this example was an incorrect ServerPort value in the client's pct.ini file. It did not match the ServerPort value defined in the server's pct.ini file.

If you receive errors such as this via the PCT tool, do not go any further with problem determination within DB2 until you have rectified these network problems, since these problems are occurring external to DB2.

Creating a loopback connection - example

When you catalog a connection to a local database as if it were a remote database, this is sometimes referred to as a "loop back". You may wish to create such a connection temporarily if you are investigating a connectivity problem. It may also be necessary if you have a multi-threaded application which connects to a local database on AIX and it is running out of shared memory segments (resulting in intermittent SQL1224N error messages).

What follows is an example illustrating how to create a loopback connection to the SAMPLE database.

1. Examine your database directory.

   Find the SAMPLE database entry in the output from the command "db2 list db directory".

   ```
   ...
   Database X entry:
   Database alias = SAMPLE
   Database name = SAMPLE
   Database drive = C:\DB2
   Database release level = a.00
   Comment =
   Directory entry type = Indirect
   Catalog node number = 0
   Alternate server hostname           =
   Alternate server port number        =
   ...
   ```

   When you have finished, there will be another catalog listed here as remote, but ultimately pointing to the same local database. There are methods of performing loopback catalogs such that the loopback alias name matches what the original database name was, however those methods will not be used in this example. Those steps are often followed when an application will be using the loopback catalog permanently, and you do not want to have to change the way the application or script was referring to the database. We are keeping it simple for the sake of this example.

2. Determine which TCP/IP port number your instance is listening on for incoming requests. To verify the port number or service name that the instance is listening on for incoming requests, issue the command ″db2 get dbm cfg″.

   ```
   ...
   TCP/IP Service name                         (SVCENAME) = db2c_DB2
   ...
   ```

   This information, along with the hostname or IP address of this server, is all that you need in order to catalog the node.

   Note: If you are creating a loopback connection in order to investigate a client connectivity issue, then if the client has cataloged the server using an IP address rather than a hostname, then you should do the same in your loopback catalog for consistency.

3. Configure the loopback connection

   a. Catalog the loopback node using the following command:

      ```
      db2 catalog tcpip node <new name> remote <hostname/ip address> server <service name/port nu
      ```

      The port number used in this command must be the same port number as the instance is listening on for incoming requests(as determined in step 2). If a service name is used, it must map to the same port number value as the instance is using.

      For example:

      ```
      db2 catalog tcpip node loopnode remote myhostname server db2c_DB2
      db2 terminate
      db2 list node directory
      ...

      Node X entry:
       Node name                      = LOOPNODE
       Comment                        =
       Directory entry type           = LOCAL
       Protocol                       = TCPIP
       Hostname                       = myhostname
       Service name                   = db2cDB2
      ...
      ```

   b. Catalog the loopback database, using the following command:

      ```
      db2 catalog db <database name> as <loop back db alias> at node <loop back node name, per ab
      ```

      For example:

      ```
      db2 catalog db sample as loopsamp at node loopnode
      db2 terminate
      db2 list db directory

      Database X entry:

       Database alias                       = LOOPSAMP
       Database name                        = SAMPLE
       Node name                            = LOOPNODE
       Database release level               = a.00
       Comment                              =
       Directory entry type                 = Remote
       Catalog database partition number    = -1
       Alternate server hostname            =
       Alternate server port number         =

      Database Y entry:

       Database alias                       = SAMPLE
       Database name                        = SAMPLE
       Database drive                       = D:\DB2
       Database release level               = a.00
      ```

```
   Comment                              =
   Directory entry type                 = Indirect
   Catalog database partition number    = 0
   Alternate server hostname            =
   Alternate server port number         =
```

4. Test the loop-back connection:

   To connect to the loopback database, enter:

   ```
   db2 connect to <loop back db alias> user <userid>
   ```

   You will be prompted for a password.

   For example:

   ```
   C:>db2 connect to loopsamp user tester
   Enter current password for tester:

      Database Connection Information

    Database server        = DB2/NT 8.2.0
    SQL authorization ID   = TESTER
    Local database alias   = LOOPSAMP
   ```

5. If you wish to remove this node and database catalog after you have completed your tests, you can do so as follows:

   ```
   db2 uncatalog node loopnode
   db2 uncatalog db loopsamp
   db2 terminate
   ```

Confirming TCP/IP configuration for a DB2 instance

Configuration errors are commonly the cause if your communication problem has the following characteristics:

- It is not an intermittent error;
- The connection has either never worked, or has changed recently;
- You can connect successfully locally;
- You tried creating a loopback connection on the server and it failed. (In this situation, you should continue to debug the problem within the loopback connection on the server. Once you have resolved the issue there, you will know what changes need to be made on the client (if any)).

Verifying the TCP/IP setup on the server

The steps to confirm the setup on the server are as follows:

1. Verify the existence of the database by issuing one of the following commands:
   - `db2 list db directory`
   - `db2 list db directory show detail`

     If this is a multi-tier environment, you may see the database cataloged as a remote database.

     ```
     Database X entry:

     Database alias = SAMPLE
     Database name = SAMPLE
     Database drive = C:\DB2
     Database release level = a.00
     Comment =
     Directory entry type = Indirect
     Catalog node number = 0
     Alternate server hostname           =
     Alternate server port number        =
     ```

2. Verify that the instance is configured to start listening on the appropriate protocol, via the command `db2set -all`. Look for the parameter DB2COMM.

```
C:\>db2set -all

[e] DB2PATH=D:\IBM\SQLLIB
[i] DB2INSTOWNER=DB2INST1
[i] DB2PORTRANGE=60000:60003
[i] DB2INSTPROF=D:\IBM\SQLLIB
[i] DB2COMM=TCPIP
[g] DB2_EXTSECURITY=YES
[g] DB2SYSTEM=LISAC02
[g] DB2PATH=D:\IBM\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS00
```

3. Verify that the appropriate protocol-specific parameters are set in the database manager configuration settings, so that DB2 knows what values to listen on. For example, issue `db2 get dbm cfg` and look in the following section:

```
...
NetBIOS Workstation name (NNAME) =
TCP/IP Service name (SVCENAME) = db2cDB2
...
```

4. If the previous step resulted in a service name instead of a port number for the SVCENAME field in your environment, then confirm that the value listed there is mapped to a unique port number in the operating system's /etc/services file. For example:

```
DB2_DB2         60000/tcp #These ports are reserved for the DB2 Fast Communications Manager
DB2_DB2_1       60001/tcp
DB2_DB2_2       60002/tcp
DB2_DB2_END     60003/tcp
db2c_DB2 50000/tcp #This is the connection port for instance DB2
```

Starting DB2 communication listeners on the server

1. Create a backup copy of the db2diag.log, then clear the contents of the original db2diag.log file. Raise the diagnostic level to 4, then stop and restart this DB2 instance, via the following commands:

```
db2 update dbm cfg using diaglevel 4
```

Note: You must refer to the db2diag.log instead of the notification log in this situation, because the notification log does not contain this level of detail even when using notification level 4.

2. Verify that the db2diag.log shows that the listeners for the specific protocol which we are interested have been started successfully.

```
2005-02-11-15.59.58.828000-480 I2036155H322       LEVEL: Info
PID     : 2536                 TID  : 2632        PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000
FUNCTION: DB2 UDB, common communication, sqlcctcp_start_listen, probe:80
MESSAGE : DIA3000I "TCPIP" protocol support was successfully started.
```

3. If you had not performed one of the necessary setup steps, you would receive the following error when you performed the db2start command:

```
SQL5043N Support for one or more communications protocols
failed to start successfully. However, core database manager
functionality started successfully.
```

4. This would be accompanied by specific error messages in the db2diag.log file if DIAGLEVEL was set to 4 at the time. For example, if you I had not set the SVCENAME parameter in the database manager configuration settings, you would receive the following error in the db2diag.log:

```
2005-02-11-16.14.05.843000-480 E2206H457          LEVEL: Error
PID     : 2516                 TID  : 2204        PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000
FUNCTION: DB2 UDB, common communication, sqlcctcpconnmgr, probe:50
```

```
MESSAGE : ADM7006E  The SVCENAME DBM configuration parameter was not
          configured.  Update the SVCENAME configuration parameter using the
          service name defined in the TCP/IP services file.
```

5. Once you have confirmed that the DB2 protocol-specific listeners are started successfully for the instance, don't forget to downgrade the diagnostic level, so that you do not suffer performance problems. Returning it to the default diagnostic level of 3 is done as follows:

   ```
   db2 update dbm cfg using diaglevel 3
   ```

Verifying the TCP/IP setup on the client

The steps to confirm the setup on the client are as follows:

1. Confirm the values that you used in your cataloging of the remote server's instance by issuing one of the following commands:
   - db2 list node directory
   - db2 list node directory show detail

   ```
   C:\>db2 list node directory show detail
   ...
   Node 1 entry:
    Node name                     = MYNODE
   Comment                       =
   Directory entry type          = LOCAL
   Protocol                      = TCPIP
   Hostname                      = myserver
   Service name                  = 50000
   Remote instance name          =
   System                        =
   Operating system type         = None
   ...
   ```

   The "Service name" value in this output should match the port number defined in the server instance's database manager configuration SVCENAME setting. If a service name is used instead of a port number, it is important that the port number to which the service name on the client maps (in the client's etc/services file) should match the port number used on the server.
2. Verify that you can ping the Hostname value exactly as it appears in the results from step (1).
3. Another test that you can perform is to use the telnet command to check whether there is something (in this case, hopefully a DB2 instance) listening on a particular port number on a particular server. For example: `telnet myserver.ibm.com 50000`. Telnet does not have to be enabled on the server. If there is indeed something listening on that port then you should see that the telnet window will open, but then hang. This means that you have indeed reached the server, and that something is indeed listening on that port. If, on the other hand, you receive an immediate error, then either:
   a. You have used the wrong hostname or IP address values. If you used a hostname here, try using the IP address, in case it is a hostname resolution problem;
   b. You have used the wrong service name or port number. If you used a service name, try using the port number instead, in case the service name was not mapped correctly in the `/etc/services` file on this client;
   c. There is a firewall between the client and server, and it is not allowing communication on this port. Verify with your network administrator whether or not this is the case;
   d. DB2 on the server is not listening on this port. Refer back "Verification of the setup on the server" in order to confirm the setup of the server.

4. If the telnet and ping tests worked, confirm that the proper database values appear in the database and DCS catalogs, as appropriate. Errors in these catalogs do not usually result in communication errors, however.

At this point you have confirmed that the configuration of both the client and the server are correct from the DB2 point of view. If you continue to receive a communication error, these are situations where you should pursue obtaining diagnostics like db2 traces.

Related reference:

- Configuring TCP/IP communications for a DB2 instance
- Verifying port range availability on participating computers (Windows)
- Enabling communications between database partition servers (UNIX)

Confirming NetBIOS configuration for a DB2 instance

Configuration errors are commonly the cause if your communication problem has the following characteristics:

- It is not an intermittent error;
- The connection has either never worked, or has changed recently;
- You can connect successfully locally;
- You tried creating a loopback connection on the server and it failed. (In this situation, you should continue to debug the problem within the loopback connection on the server. Once you have resolved the issue there, you will know what changes need to be made on the client (if any)).

Verifying the NetBIOS setup on the server

The steps to confirm the setup on the server are as follows:

1. Verify the existence of the database by issuing one of the following commands:
   - `db2 list db directory`
   - `db2 list db directory show detail`

   If this is a multi-tier environment, you may see the database cataloged as a remote database.

   ```
   Database X entry:
   Database alias = SAMPLE
   Database name = SAMPLE
   Database drive = C:\DB2
   Database release level = a.00
   Comment =
   Directory entry type = Indirect
   Catalog node number = 0
   Alternate server hostname           =
   Alternate server port number        =
   ```

2. Verify that the instance is configured to start listening on the appropriate protocol, via the command `db2set -all`. Look for the parameter `DB2COMM`.

   ```
   C:\>db2set -all

   [e] DB2PATH=D:\IBM\SQLLIB
   [i] DB2INSTOWNER=DB2INST1
   [i] DB2PORTRANGE=60000:60003
   [i] DB2INSTPROF=D:\IBM\SQLLIB
   [i] DB2COMM=NETBIOS
   [g] DB2_EXTSECURITY=YES
   [g] DB2SYSTEM=LISAC02
   [g] DB2PATH=D:\IBM\SQLLIB
   [g] DB2INSTDEF=DB2
   [g] DB2ADMINSERVER=DB2DAS00
   ```

3. Verify that the appropriate protocol-specific parameters are set in the database manager configuration settings, so that DB2 knows what values to listen on. For example, issue `db2 get dbm cfg` and look in the following section:

   ```
   ...
   NetBIOS Workstation name (NNAME) = server1
   ...
   ```

   Note: The NNAME must be unique among all NetBIOS nodes in the network. The maximum length of the nname is 8 characters.

Starting DB2 communication listeners on the server:

1. Create a backup copy of the db2diag.log, then clear the contents of the original db2diag.log file. Raise the diagnostic level to 4via the following commands:

   ```
   db2 update dbm cfg using diaglevel 4
   ```

   Note: You must refer to the db2diag.log instead of the notification log in this situation, because the notification log does not contain this level of detail even when using notification level 4.
2. Verify that the db2diag.log shows that the listeners for the specific protocol which we are interested have been started successfully.

   ```
   2005-02-11-16.51.44.640000-480 I2987H328         LEVEL: Info
   PID     : 2624                 TID  : 2464       PROC : db2syscs.exe
   INSTANCE: DB2                  NODE : 000
   FUNCTION: DB2 UDB, common communication, sqlccnb_start_listen, probe:62
   MESSAGE : DIA3001E "NETBIOS" protocol support was successfully started.
   ```
3. If you had not performed one of the necessary setup steps, you would receive the following error when you performed the `db2start` command:

   ```
   SQL5043N Support for one or more communications protocols
   failed to start successfully. However, core database manager
   functionality started successfully.
   ```
4. This would be accompanied by specific error messages in the db2diag.log file if `DIAGLEVEL` was set to 4 at the time. For example, if you had not set the `SVCENAME` parameter in the database manager configuration settings, you would receive the following error in the db2diag.log:

   ```
   2005-02-11-17.04.18.750000-480 I1906H457         LEVEL: Error
   PID     : 2488                 TID  : 2628       PROC : db2syscs.exe
   INSTANCE: DB2                  NODE : 000
   FUNCTION: DB2 UDB, common communication, sqlccnbconnmgr_child, probe:34
   MESSAGE : DIA3426C "NETBIOS" protocol support: Workstation name (nname) for
             this server is NOT valid.  Enter a valid Workstation name (nname) in
             the Database Manager Configuration.
   ```
5. Once you have confirmed that the DB2 protocol-specific listeners are started successfully for the instance, don't forget to downgrade the diagnostic level, so that you do not suffer performance problems. Returning it to the default diagnostic level of 3 is done as follows:

   ```
   db2 update dbm cfg using diaglevel 3
   ```

Verifying the NetBIOS setup on the client

The steps to confirm the setup on the client are as follows:

1. Verify that an appropriate NetBIOS Workstation name (NNAME) value is set in the database manager configuration settings on the client. For example, issue db2 get dbm cfg and look in the following section:

   ```
   ...
   NetBIOS Workstation name (NNAME) = client1
   ...
   ```

Note: The NNAME must be unique among all NetBIOS nodes in the network. The maximum length of the nname is 8 characters.

2. Verify the local adapter that will be used for the connection. Refer to Determining the logical adapter number of the client for the NetBIOS connection (Windows).
3. Confirm the values that you used when cataloging the remote server's instance by issuing one of the following commands:
   - `db2 list node directory`
   - `db2 list node directory show detai`

```
C:\>db2 list node directory
...
Node 1 entry:

 Node 3 entry:

 Node name                     = NETNODE
 Comment                       =
 Directory entry type          = LOCAL
 Protocol                      = NETBIOS
 Adapter number                = 0
 Server NNAME                  = SERVER1
...
```

The ″`Adapter number`″ entry should indicate the client's logical adapter number (as determined in step 2). The ″`Server NNAME`″ entry should indicate the server's Workstation name (nname, as specified in the server's database manager configuration file).

At this point you have confirmed that the configuration of both the client and the server are correct from the DB2 point of view. If you continue to receive a communication error, these are situations where you should pursue obtaining diagnostics like db2 traces.

Related reference:

- Configuring NetBIOS communications on the client using the CLP
- NetBIOS parameter values worksheet
- nname - NetBIOS workstation name configuration parameter

Testing a database connection using the Configuration Assistant

Click on this link for more information.

Common connectivity problems

Initial connection mainframe or midrange server is not successful

Click on this link for more information on ″Initial connection mainframe or midrange server is not successful″.

Problems encountered after an initial connection

Click on this link for more information on ″Problems encountered after an initial connection″.

Common DB2 Connect problems

Database crashes

In order to investigate these issues, you need to understand and distinguish the difference between a database crash and an application crash.

Determine if the database instance has crashed:

To determine if the database instance has crashed the administration notification log at various probe points must be examined. In this file, you may see the following:

```
2005-01-29-03.13.07.166360   Instance:db2inst1   Node:000
PID:1310914(db2agent (BIDB) 0)   TID:1   Appid:*N0.db2inst2.050128190001
oper system services  sqloEDUCodeTrapHandler Probe:10   Database:SAMPLE

ADM0503C  An unexpected internal processing error has occurred.  ALL DB2
PROCESSES ASSOCIATED WITH THIS INSTANCE HAVE BEEN SHUTDOWN.  Diagnostic  information has been recorde
```

The db2diag.log would show additional information, as follows:

```
2005-01-20-11.15.04.507340-360 E5806002A512       LEVEL: Severe
PID     : 282632               TID  : 1           PROC : db2agntp 0
INSTANCE: db2inst1            NODE : 000
FUNCTION: DB2 UDB, oper system services, sqloEDUCodeTrapHandler,
probe:10
MESSAGE : ADM0503C  An unexpected internal processing error has
occurred.  ALL DB2 PROCESSES ASSOCIATED WITH THIS INSTANCE HAVE BEEN
SHUTDOWN.  Diagnostic information has been recorded.  Contact IBM Support for further assistance.

2005-01-20-11.15.04.508711-360 E5806515A645       LEVEL: Severe
PID     : 282632               TID  : 1           PROC : db2agntp 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, oper system services, sqloEDUCodeTrapHandler,
probe:20
DATA #1 : Signal Number Recieved, 4 bytes
11
DATA #2 : Siginfo, 64 bytes
0x0FFFFFFFFFFFE050 : 0000 000B 0000 0000 0000 0032 0000 0000
..........2....
0x0FFFFFFFFFFFE060 : 0000 0000 0000 0000 0000 0001 1170 A370
............p.p
0x0FFFFFFFFFFFE070 : 0000 0000 0000 0000 0000 0000 0000 0000
...............
0x0FFFFFFFFFFFE080 : 0000 0000 0000 0000 0000 0000 0000 0000
...............
```

In this example, second db2diag.log entry shows that the function `sqloEDUCodeTrapHandler` has returned a Signal number of 11. This means that the DB2 signal handler has caught a signal #11. On the UNIX platform the header file called signal.h is usually located in `/usr/include/sys`. In this example you will determine that a signal #11 is a segmentation violation (SIGSEGV):

Extract of the signal.h header file

```
...
#define SIGBUS 10 /* (*) bus error (specification exception) */
#define SIGSEGV 11 /* (*) segmentation violation */
#define SIGSYS 12 /* (*) bad argument to system call */
...
```

This is the first indication that the database has indeed crashed due to a segmentation violation and the database signal handler has caught the signal. The next step is to determine the process ID (PID) that has crashed. We return to the db2diag.log file, to find the "abnormally terminated process":

```
2005-01-20-11.15.04.558786-360 I5807161A433       LEVEL: Severe
PID     : 2027586              TID  : 1           PROC : db2gds 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, oper system services, sqloEDUSIGCHLDHandler, probe:50
DATA #1 : String, 160 bytes
Detected the death of an EDU with process id 282632
The signal number that terminated this process was 11
Look for trap files (t282632.*) in the dump directory
```

The function sqloEDUSIGCHLDHandler at probe 50 has provided the process id of the problematic EDU, and the name of the trap file to reference. For this example you will get a file called t282632.000 in the DIAGPATH directory. On some platforms such as AIX, a CORE file may be generated as well.

The trap file contains a stack traceback of all the functions on the stack for the process that crashed. For more information about how to interpret the stack trace, and how to search for appropriate APARs, refer to Analyzing the stack trace back

Related reference:

- Location of logs and files
- Trap files
- Common signals and exceptions that cause trap file generation

db2dart and INSPECT–Database inspection tools

The db2dart and inspect tools can be used in database crash situations to ensure that the databases have not been corrupted. An overview of each of these tools follows.

Overview of the db2dart tool:

db2dart is a command which can be used to verify the architectural correctness of databases and the objects within them. It can also be used to display the contents of database control files in order to extract data from tables that might otherwise be inaccessible.

To display all of the possible options, simply execute the db2dart utility without any parameters. Some options that require parameters, such as the table space ID, are prompted for if they are not explicitly specified on the command line.

By default, db2dart will create a report file with the name databaseName.RPT. For single-partition database environments, the file is created in the current directory. For multiple-partition database environments, the file is created under a subdirectory in the diagnostic directory. The subdirectory is called DART####, where #### is the partition number.

db2dart accesses the data and metadata in a database by reading them directly from disk. Because of that, db2dart should never be run against a database that still has active connections. If there are connections, db2dart will not know about pages in the buffer pool, control structures in memory, etc. and may report false errors as a result. Similarly, if you run db2dart against a database that requires crash recovery or that has not completed roll-forward recovery, similar inconsistencies might result due to the inconsistent nature of the data on disk.

Inspecting databases, table spaces, and tables via db2dart:

The default behaviour for db2dart is to inspect the entire database. Only the database name must be provided in this case. For example:

```
C:\>db2dart sample

         The requested DB2DART processing has completed successfully!
                        Complete DB2DART report found in:
C:\IBM\SQLLIB\DB2\DART0000\SAMPLE.RPT

Complete DB2DART report found in: SAMPLE.RPT
```

As the output states, the full db2dart report can be found in the file SAMPLE.RPT. You will also notice that in this case db2dart did not find any problems with the database.

If a database is very large and you are only interested in one table space, you can use the /TS option. When using this option, you must either provide the table space ID on the command line (by specifying the /TSI parameter) or you can let db2dart prompt you for it. If you do not know the table space ID, you can obtain it via the command ″DB2 LIST TABLESPACES″ command. For example, to inspect the USERSPACE1 table space (which has a table space ID of 2 in the sample database), either of these commands will work:

```
db2dart sample /ts /tsi 2
db2dart sample /ts <= When prompted for the table space ID, enter "2".
```

Similarly, a single table and its associated objects (LOBs, indexes, etc.) can be inspected using the /T option. When using this option, you must provide either the table name or object ID and the ID of the table space in which the table resides. To determine the object ID and table space ID for a table, you can query the FID and TID columns of the SYSIBM.SYSTABLES catalog table. For example, determine the object ID and table space ID for the EMP_PHOTO table in the sample database by executing the following query:

```
C:\>db2 connect to sample
Database server          = DB2/NT 8.2.0
 SQL authorization ID   = LISAC
 Local database alias   = SAMPLE

C:\>db2 "select creator,name,tid,fid from sysibm.systables where name =
'EMP_PHOTO'"
CREATOR NAME TID FID
------------------ -------------------- ------ ------
DB2 EMP_PHOTO 2 8
1 record(s) selected.

C:\>db2 connect reset
DB20000I The SQL command completed successfully.
```

To inspect this table, execute either of the following db2dart commands:

```
db2dart sample /t /tsi 2 /oi 8
db2dart sample /t <= When prompted for the table ID and table space ID, enter "8 2".
```

As mentioned above, the table name can be specified instead of the object ID:

```
db2dart sample /t /tsi 2 /tn EMP_PHOTO
db2dart sample /t <= When prompted for the table name and table space ID,enter "EMP_PHOTO 2".
```

Dumping formatted table data via db2dart:

If a table space or table becomes corrupt for any reason (for example due to a bad disk or disk controller), attempts to access the table through SQL may not work. (The SQL statement may fail with an error or the database may be marked bad and all connections will be dropped.) In such a case, entries will likely be written to the db2diag.log file, indicating that a bad page was encountered.

```
2004-10-12-16.49.20.119228+120 I3292G436           LEVEL: Error
PID     : 14974                TID  : 605992128     PROC : db2bm.14206.5
INSTANCE: db2inst1             NODE : 000           DB   : SAMPLE
APPHDL  : 0-777                APPID: *LOCAL.db2inst1.000
FUNCTION: DB2 UDB, buffer pool services, sqlbrdpg, probe:1143
RETCODE : ZRC=0x86020001=-2046689279=SQLB_BADP "page is bad"
          DIA8400C A bad page was encountered.
```

If you see such entries, you should run db2dart against the database (or table space) to determine the extent of the damage.

If this happens, it may be necessary to extract all of the data possible so that the table space and table can be rebuilt. In such a situation, the `/DDEL` option of db2dart can be used to extract the table data and place it into a delimited ASCII file. Note that due to the nature of ASCII files, some columns (such as LOB columns) cannot be extracted from the table. db2dart will tell you if this is the case.

When using the `/DDEL` option, you must provide a table space ID, object ID, starting page number, and number of pages. To extract all of the pages, use 0 for the starting page number and some very large number for the number of pages. (Specifying more pages than actually exist will not cause any problems.)

The ORG table in the sample database resides in table space 2 and has an object ID of 2. To extract all of the data from this table, execute this command:

```
db2dart sample /ddel
```

When prompted, enter either of the following lines of input:

```
2 2 0 1000
ORG 2 0 1000
```

You will then be presented with the column definitions for the table and will be asked to specify an output file name:

```
Table object data formatting start.
Please enter
Table ID or name, tablespace ID, first page, num of pages:
(suffic page number with 'p' for pool relative)
2 2 0 1000
5 of 5 columns in the table will be dumped.
Column numbers and datatypes of the columns dumped:
0 SMALLINT
1 VARCHAR() -VARIABLE LENGTH CHARACTER STRING
2 SMALLINT
3 VARCHAR() -VARIABLE LENGTH CHARACTER STRING
4 VARCHAR() -VARIABLE LENGTH CHARACTER STRING
Default filename for output data file is TS2T2.DEL,
do you wish to change filename used? y/n
```

You can choose the default or specify a new one. The output file will be created in the current directory be default.

When the extraction is complete, you will see output as follows:

```
  Filename used for output data file is TS2T2.DEL.  If existing file, data will
be appended to it.
  Formatted data being dumped ...
```

```
Dumping Page 0 ....
Table object data formatting end.
      The requested DB2DART processing has completed successfully!
```

Overview of the INSPECT command:

An online inspection command called INSPECT was introduced in DB2 Version 8. This command allows you to perform similar database, table space, and table checking as is done by db2dart. There are many benefits to using the INSPECT command including the ability to run it while there are active connections against the database. Also, it is built into the engine which means that significant performance gains are achieved through the use of buffer pools and prefetchers.

Feature comparison of DB2DART and INSPECT:

Table 2. Feature Comparison of DB2DART vs. INSPECT

Test Performed	**DB2dart V8**		**Inspect V8**	
SMS	**Offline**	**Online**	**Offline**	**Online**
Check Tablespace Files	✓			
DMS				
Extent maps pointed at by more than 1 object	✓			
Extent Maps Agree with Space Maps	✓			
For Space Map check every page for consistency bit errors				✓
For Space Map check buffer pool fields in page header				✓
Extent Map consistency bit errors				✓
Extent Map consistency bit errors				✓
Data Object				
Consistency bits error	✓			✓
Buffer pool fields in page header	✓			✓
Data Manager Fields in page header	✓			✓

Table 2. Feature Comparison of DB2DART vs. INSPECT (continued)

Test Performed	**DB2dart V8**		**Inspect V8**	
SMS	**Offline**	**Online**	**Offline**	**Online**
Each Row record type and length	✓			✓
Rows not overlapping	✓			✓
Special control rows contents	✓			
Variable length columns length and position	✓			
LF/LOB descriptors in table rows	✓			
Summary Total Pages, Used Pages, Free Space Percentage				✓

Test Performed	**DB2dart V8**		**Inspect V8**	
Index Object - each page	**Offline**	**Online**	**Offline**	**Online**
Consistency bits error	✓			✓
Buffer pool fields in page header	✓			✓
Index Manager fields in page header	✓			✓
Index key location, length, overlapping	✓			✓
Index ordering of keys	✓			
Uniqueness of unique keys	✓			
Summary Total Pages, Used Pages				✓
Block Map Object				
Check consistency bit errors	✓			✓
Buffer pool pages in Page Header	✓			✓

Test Performed	**DB2dart V8**		**Inspect V8**	
Index Object - each page	**Offline**	**Online**	**Offline**	**Online**
Block Manager fields in Page Header	✓			✓
Summary Total Pages, Used Pages				✓
Long Field Object				
Allocation structures	✓			✓
LOB Object				
Allocation Structures	✓			✓
Summary Total Pages, Pages Used				✓
Miscellaneous				
Format & Dump data pages	✓			
Format & dump index pages	✓			
Format data rows to delimited ASCII	✓			
Mark an index invalid	✓			
Initialize a data page to empty	✓			
Block Map Object	Offline	Online	Offline	Online
Check consistency bits error	✓			
Check buffer pool fields in page header	✓			
Check block manager in page header	✓			

Related reference:

- db2dart - Database Analysis and Reporting Tool Command
- INSPECT command
- First failure data capture locations

Bringing an OFFLINE table space ONLINE

Sometimes a problem will cause a table space to go OFFLINE. After fixing the problem, perform the following steps to return the state to ONLINE:

Prerequisites:

You will need the ability to disconnect all applications from the database and to issue the ALTER TABLESPACE command.

Procedure:

1. Disconnect all applications and reconnect to the database. You will see that the table space can be taken out of the OFFLINE state.
2. Use the ALTER TABLESPACE ... SWITCH ONLINE statement to bring the table space up while the rest of the database is still up and deployed.
   - If the table space can be brought up successfully after issuing the command, or if the table space was not in the OFFLINE state to begin with, DB2 will return an SQLCODE of 0.
   - If the table space cannot be brought up successfully because there are still problems with one or more of the containers, DB2 will return an SQLCODE of —293. You can force the database to restart by using the RESTART ... DROP PENDING TABLESPACE, but will have to drop any faulty table spaces afterwards.

Related reference:

- "ALTER TABLESPACE statement" in the *SQL Reference, Volume 2*
- "RESTART DATABASE Command" in the *Command Reference*

Data movement issues

This section will help you identify the source of common load, import or export problems.

Loading data

`LOAD` is a utility used to bulk-load data from a file or pipe into a database. `LOAD` consists of up to three phases. The load phase loads the data, the build phase builds the indexes if there are any, and the delete phase deletes duplicates if there are unique indexes.

The main load processes are as follows: (X is a number identifying one of many)

- `db2agent`
- `db2lmr Load Media Reader`. This process reads the input.
- `db2lbmX Load Buffer Manipulator`. Writes loaded data to the database.
- `db2lfrmX Load Formatter`. Formats the input data into internal form.
- `db2lrid Ridder`. Organises data to be written to disk. Performs the index sort.
- `db2lmwX Load Media Writers`. Write the load copy.

LOAD failure:

What do you do when a `LOAD` fails and the table space is not accessible? The following options are available.

- Restart the `LOAD`

- Terminate the LOAD
- Restore from the most recent backup and rollforward to a point-in-time before the failed LOAD
- Drop and rebuild the table space.

A typical message for a failed load in the db2diag.log is as follows:

```
2004-02-17-11.26.29.021820 Instance:db2inst1 Node:000
PID:51945658(db2lload 5) TID:1 Appid:*N17.db2p001.0E00B7172332
database utilities call_sqluvload Probe:40 Database:SAMPLE

Load Error: Load failed for table DB2INST1.ORG in tablespace 1.
```

In addition, you can issue the following command to examine the state of the table space whose table is being loaded into:

```
db2 list tablespaces show detail
```

Typical output is as follows:

```
Tablespace ID = 2
Name = USERPACE1
Type = System managed space
Contents = Any data
State = 0x0004
Detailed explanation:
 Quiesced: EXCLUSIVE
...
Number of quiescers = 1
 Quiescer 1:
 Tablespace ID = 2
 Object ID = 3
```

Use the db2tbst tool that is provided with DB2 to determine that state of the table space.

For the above example, you would issue the following command

```
db2tbst 0x0004
```

This returns:

```
State = Quiesced Exclusive
```

In this state, the table space is inaccessible. To remove the table space from this state, issue:

```
db2 quiesce tablespaces for table staff reset
```

where STAFF is the name of the table being loaded into.

Prior to Version 8, load required exclusive access to table spaces that contained objects belonging to the table being loaded. In Version 8, load operates at the table level and no longer requires exclusive access to the table space. The load utility does not quiesce the table spaces involved in the load operation, and uses table space states only for load operations with the COPY NO option specified. Refer to Load overview.

Bringing a table back online:

Load also uses the table space state LOAD PENDING while in the load or build phase, or DELETE PENDING during the delete phase. These states persist after the load is interrupted or the load fails. They can only be released by performing a LOAD RESTART, a LOAD TERMINATE, or a LOAD REPLACE. See Pending states after a load operation.

:

- Load overview
- Pending states after a load operation
- db2tbst - Get Tablespace State Command
- Load temporary files
- Load utility log records

Message files from IMPORT, EXPORT, and LOAD

Click on this link for more information.

Extender issues

DB2 Spatial and Geodetic Extenders

How to interpret DB2 Spatial Extender messages

You can work with DB2® Spatial Extender through four different interfaces:

- DB2 Spatial Extender stored procedures
- DB2 Spatial Extender functions
- DB2 Spatial Extender Command Line Processor (CLP)
- DB2 Control Center

All interfaces return DB2 Spatial Extender messages to help you determine whether the spatial operation that you requested completed successfully or resulted in an error.

The following table explains each part of this sample DB2 Spatial Extender message text:

```
GSE0000I: The operation was completed successfully.
```

Table 3. The parts of the DB2 Spatial Extender message text

Message text part	Description
GSE	The message identifier. All DB2 Spatial Extender messages begin with the three-letter prefix GSE.
0000	The message number. A four digit number that ranges from 0000 through 9999.
I	The message type. A single letter that indicates the severity of message: **C** Critical error messages **N** Non-critical error messages **W** Warning messages **I** Informational messages
The operation was completed successfully.	The message explanation.

The explanation that appears in the message text is the brief explanation. You can retrieve additional information about the message that includes the detailed explanation and suggestions to avoid or correct the problem. To display this additional information:

1. Open an operating system command prompt.
2. Enter the DB2 help command with the message identifier and message number to display additional information about the message. For example:

   ```
   DB2 "? GSEnnnn"
   ```

   where *nnnn* is the message number.

You can type the GSE message identifier and letter indicating the message type in uppercase or lowercase. Typing DB2 ″? GSE0000I″ will yield the same result as typing db2 ″? gse0000i″.

You can omit the letter after the message number when you type the command. For example, typing DB2 ″? GSE0000″ will yield the same result as typing DB2 ″? GSE0000I″.

Suppose the message code is GSE4107N. When you type DB2 ″? GSE4107N″ at the command prompt, the following information is displayed:

```
GSE4107N Grid size value "<grid-size>" is not valid where it is used.

Explanation: The specified grid size "<grid-size>" is not valid.

 One of the following invalid specifications was made when the grid index
 was created with the CREATE INDEX statement:

  -  A number less than 0 (zero) was specified as the grid size for the
     first, second, or third grid level.

  -  0 (zero) was specified as the grid size for the first grid level.

  -  The grid size specified for the second grid level is less than the grid
     size of the first grid level but it is not 0 (zero).

  -  The grid size specified for the third grid level is less than the grid
     size of the second grid level but it is not 0 (zero).

  -  The grid size specified for the third grid level is greater than 0 (zero)
     but the grid size specified for the second grid level is 0 (zero).

User Response: Specify a valid value for the grid size.

msgcode: -4107

sqlstate: 38SC7
```

If the information is too long to display on a single screen and your operating system supports the **more** executable program and pipes, type this command:

```
db2 "? GSEnnnn" | more
```

Using the **more** program will force the display to pause after each screen of data so that you can read the information.

DB2 Spatial Extender stored procedure output parameters

DB2® Spatial Extender stored procedures are invoked *implicitly* when you enable and use Spatial Extender from the DB2 Control Center or when you use the DB2

Spatial Extender CLP (db2se). You can invoke stored procedures *explicitly* in an application program or from the DB2 command line.

This topic describes how to diagnose problems when stored procedures are invoked explicitly in application programs or from the DB2 command line. To diagnose stored procedures invoked implicitly, you use the messages returned by the DB2 Spatial Extender CLP or the messages returned by the DB2 Control Center. These messages are discussed in separate topics.

DB2 Spatial Extender stored procedures have two output parameters: the message code (msg_code) and the message text (msg_text). The parameter values indicate the success or failure of a stored procedure.

msg_code

The msg_code parameter is an integer, which can be positive, negative, or zero (0). Positive numbers are used for warnings, negative numbers are used for errors (both critical and non-critical), and zero (0) is used for informational messages.

The absolute value of the msg_code is included in the msg_text as the message number. For example

- If the msg_code is 0, the message number is 0000.
- If the msg_code is -219 , the message number is 0219. The negative msg_code indicates that the message is a critical or non-critical error.
- If the msg_code is +1036, the message number is 1036. The positive msg_code number indicates that the message is a warning.

The msg_code numbers for Spatial Extender stored procedures are divided into the three categories shown in the following table:

Table 4. Stored procedure message codes

Codes	Category
0000 - 0999	Common messages
1000 - 1999	Administrative messages
2000 - 2999	Import and export messages

msg_text

The msg_text parameter is comprised of the message identifier, the message number, the message type, and the explanation. An example of a stored procedure msg_text value is:

```
GSE0219N   An EXECUTE IMMEDIATE statement
           failed. SQLERROR = "<sql-error>".
```

The explanation that appears in the msg_text parameter is the brief explanation. You can retrieve additional information about the message that includes the detailed explanation and suggestions to avoid or correct the problem.

For a detailed explanation of the parts of the msg_text parameter, and information on how to retrieve additional information about the message, see the topic: How to interpret DB2 Spatial Extender messages.

Working with stored procedures in applications:

When you call a DB2 Spatial Extender stored procedure from an application, you will receive the msg_code and msg_text as output parameters. You can:

- Program your application to return the output parameter values to the application user.
- Perform some action based on the type of msg_code value returned.

Working with stored procedures from the DB2 command line:

When you invoke a DB2 Spatial Extender stored procedure from the DB2 command line, you receive the msg_code and the msg_text output parameters. These output parameters indicate the success or failure of the stored procedure.

Suppose you connect to a database and want to invoke the ST_disable_db stored procedure. The example below uses a DB2 CALL command to disable the database for spatial operations and shows the output value results. A force parameter value of 0 is used, along with two question marks at the end of the CALL command to represent the msg_code and msg_text output parameters. The values for these output parameters are displayed after the stored procedure runs.

```
call db2gse.st_disable_db(0, ?, ?)

  Value of output parameters
  --------------------------
  Parameter Name  : MSGCODE
  Parameter Value : 0

  Parameter Name  : MSGTEXT
  Parameter Value : GSE0000I  The operation was completed successfully.

  Return Status = 0
```

Suppose the msg_text returned is GSE2110N. Use the DB2 help command to display more information about the message. For example:

```
"? GSE2110"
```

The following information is displayed:

```
GSE2110N    The spatial reference system for the
            geomentry in row "<row-number>" is invalid.
            The spatial reference system's
            numerical identifier is "<srs-id>".

Explanation: In row row-number, the geometry that is
to be exported uses an invalid spatial reference system.
The geometry cannot be exported.

User Response: Correct the indicated geometry or
exclude the row from the export operation by
modifying the SELECT statement accordingly.

msg_code: -2110

sqlstate: 38S9A
```

DB2 Spatial Extender function messages

The messages returned by DB2® Spatial Extender functions are typically embedded in an SQL message. The SQLCODE returned in the message indicates if an error occurred with the function or that a warning is associated with the function. For example:

- The SQLCODE -443 (message number SQL0443) indicates that an error occurred with the function.
- The SQLCODE +462 (message number SQL0462) indicates that a warning is associated with the function.

The following table explains the significant parts of this sample message:

```
DB21034E  The command was processed as an SQL statement because it was
not a valid Command Line Processor command.  During SQL processing it
returned: SQL0443N  Routine "DB2GSE.GSEGEOMFROMWKT"
(specific name "GSEGEOMWKT1") has returned an error
SQLSTATE with diagnostic text "GSE3421N  Polygon is not closed.".
SQLSTATE=38SSL
```

Table 5. The significant parts of DB2 Spatial Extender function messages

Message part	Description
SQL0443N	The SQLCODE indicates the type of problem.
GSE3421N	The DB2 Spatial Extender message number and message type. The message numbers for functions range from GSE3000 to GSE3999. Additionally, common messages can be returned when you work with DB2 Spatial Extender functions. The message numbers for common messages range from GSE0001 to GSE0999.
Polygon is not closed	The DB2 Spatial Extender message explanation.
SQLSTATE=38SSL	An SQLSTATE code that further identifies the error. An SQLSTATE code is returned for each statement or row. • The SQLSTATE codes for Spatial Extender function errors are 38S*xx*, where each *x* is a character letter or number. • The SQLSTATE codes for Spatial Extender function warnings are 01HS*x*, where the *x* is a character letter or number.

An example of an SQL0443 error message:

Suppose that you attempt to insert the values for a polygon into the table POLYGON_TABLE, as shown below:

```
INSERT INTO polygon_table ( geometry )
VALUES ( ST_Polygon ( 'polygon (( 0 0, 0 2, 2 2, 1 2)) ') )
```

This results in an error message because you did not provide the end value to close the polygon. The error message returned is:

```
DB21034E  The command was processed as an SQL statement because it was
not a valid Command Line Processor command.  During SQL processing it
returned: SQL0443N  Routine "DB2GSE.GSEGEOMFROMWKT"
(specific name "GSEGEOMWKT1") has returned an error
SQLSTATE with diagnostic text "GSE3421N  Polygon is not closed.".
SQLSTATE=38SSL
```

The SQL message number `SQL0443N` indicates that an error occurred and the message includes the Spatial Extender message text `GSE3421N Polygon is not closed`.

When you receive this type of message:

1. Locate the GSE message number within the DB2 or SQL error message.
2. Use the DB2 help command (DB2 ?) to see the Spatial Extender message explanation and user response. Using the above example, type the following command in an operating system command line prompt:

```
DB2 "? GSE3421"
```

The message is repeated, along with a detailed explanation and recommended user response.

DB2 Spatial Extender CLP messages

The DB2® Spatial Extender CLP (db2se) returns messages for:

- Stored procedures, if invoked implicitly.
- Shape information, if you have invoked the **shape_info** subcommand program from the DB2 Spatial Extender CLP. These are informational messages.
- Migration operations.
- Import and export shape operations to and from the client.

Examples of stored procedure messages returned by the DB2 Spatial Extender CLP:

Most of the messages returned through the DB2 Spatial Extender CLP are for DB2 Spatial Extender stored procedures. When you invoke a stored procedure from the DB2 Spatial Extender CLP, you will receive message text that indicates the success or failure of the stored procedure.

The message text is comprised of the message identifier, the message number, the message type, and the explanation. For example, if you enable a database using the command `db2se enable_db testdb`, the message text returned by the Spatial Extender CLP is:

```
Enabling database. Please wait ...

GSE1036W   The operation was successful.  But
           values of certain database manager and
           database configuration parameters
           should be increased.
```

Likewise, if you disable a database using the command `db2se disable_db testdb` the message text returned by the Spatial Extender CLP is:

```
GSE0000I  The operation was completed successfully.
```

The explanation that appears in the message text is the brief explanation. You can retrieve additional information about the message that includes the detailed explanation and suggestions to avoid or correct the problem. The steps to retrieve this information, and a detailed explanation of how to interpret the parts of the message text, are discussed in a separate topic.

If you are invoking stored procedures through an application program or from the DB2 command line, there is a separate topic that discusses diagnosing the output parameters.

Example of shape information messages returned by the Spatial Extender CLP:

Suppose you decide to display information for a shape file named `office`. Through the Spatial Extender CLP (db2se) you would issue this command:

```
db2se shape_info -fileName /tmp/offices
```

This is an example of the information that displays:

```
Shape file information
----------------------
```

```
File code                  = 9994
File length (16-bit words) = 484
Shape file version         = 1000
Shape type                 = 1 (ST_POINT)
Number of records          = 31

Minimum X coordinate = -87.053834
Maximum X coordinate = -83.408752
Minimum Y coordinate = 36.939628
Maximum Y coordinate = 39.016477
Shapes do not have Z coordinates.
Shapes do not have M coordinates.

Shape index file (extension .shx) is present.

Attribute file information
--------------------------
dBase file code                = 3
Date of last update            = 1901-08-15
Number of records              = 31
Number of bytes in header      = 129
Number of bytes in each record = 39
Number of columns              = 3

Column Number  Column Name      Data Type       Length  Decimal

           1   NAME             C ( Character)      16        0
           2   EMPLOYEES        N (   Numeric™)      11        0
           3   ID               N (   Numeric)      11        0

Coordinate system definition: "GEOGCS["GCS_North_American_1983",
DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],
PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199432955]]"
```

Examples of migration messages returned by the Spatial Extender CLP:

When you invoke commands that perform migration operations, messages are returned that indicate the success or failure of that operation.

Suppose you invoke the migration of the database `mydb` using the command `db2se migrate mydb -messagesFile /tmp/migrate.msg`. The message text returned by the Spatial Extender CLP is:

```
Migrating database. Please wait ...
GSE0000I  The operation was completed successfully.
```

DB2 Control Center messages

When you work with DB2® Spatial Extender through the DB2 Control Center, messages will appear in the DB2 Message window. Most of the messages that you will encounter will be DB2 Spatial Extender messages. Occasionally, you will receive an SQL message. The SQL messages are returned when an error involves licensing, locking, or when a DAS service is not available. The following sections provide examples of how DB2 Spatial Extender messages and SQL messages will appear in the DB2 Control Center.

DB2 Spatial Extender messages:

When you receive a DB2 Spatial Extender message through the Control Center, the entire message text appears in the text area of DB2 Message window, for example:

```
GSE0219N   An EXECUTE IMMEDIATE statement
           failed. SQLERROR = "<sql-error>".
```

SQL messages:

When you receive an SQL message through the Control Center that pertains to DB2 Spatial Extender:

- The message identifier, message number, and message type appear on the left side of the DB2 Message window, for example: `SQL0612N`.
- The message text appears in the text area of the DB2 Message window.

The message text that appears in the DB2 Message window might contain the SQL message text and the SQLSTATE, or it might contain the message text and the detailed explanation and user response.

An example of an SQL message that contains the SQL message text and the SQLSTATE is:

```
[IBM][CLI Driver][DB2/NT] SQL0612N "<name>" is a duplicate name. SQLSTATE=42711
```

An example of an SQL message that contains the message text and the detailed explanation and user response is:

```
SQL8008N
The product "DB2 Spatial Extender" does not have a valid
license key installed and the evaluation period has expired.

Explanation:

A valid license key could not be found and the evaluation
period has expired.

User Response:

Install a license key for the fully entitled version of the
product.  You can obtain a license key for the product by
contacting your IBM® representative or authorized dealer.
```

Tracing DB2 Spatial Extender problems with the db2trc command

When you have a recurring and reproducible DB2 Spatial Extender problem, you can use the DB2 trace facility to capture information about the problem. The DB2 trace facility is activated by the **db2trc** system command. The DB2 trace facility can:

- Trace events
- Dump the trace data to a file
- Format trace data into a readable format

Restrictions:

Activate this facility only when directed by a DB2 technical support representative.

On UNIX operating systems, you must have SYSADM, SYSCTRL, or SYSMAINT authorization to trace a DB2 instance.

On Windows operating systems, no special authorization is required.

Procedure:

To trace the DB2 Spatial Extender events to memory, follow these basic steps:

1. Shut down all other applications.

2. Turn the trace on. The DB2 Support technical support representative will provide you with the specific parameters for this step. The basic command is:

   ```
   db2trc on
   ```

   Restriction: The **db2trc** command must be entered at a operating–system command prompt or in a shell script. It cannot be used in the DB2 Spatial Extender command–line interface (db2se) or in the DB2 CLP.

   You can trace to memory or to a file. The preferred method for tracing is to trace to memory. If the problem being recreated suspends the workstation and prevents you from dumping the trace, trace to a file.
3. Reproduce the problem.
4. Dump the trace to a file. For example:

   ```
   db2trc dump january23trace.dmp
   ```

   This command creates a file (*january23trace.dmp*) in the current directory with the name that you specify, and dumps the trace information in that file.

   You can specify a different directory by including the file path. For example, to place the dump file in the `/tmp/spatial/errors` directory, the syntax is:

   ```
   db2trc dump /tmp/spatial/errors/january23trace.dmp
   ```

   Dump the trace immediately after the problem occurs.
5. Turn the trace off. For example:

   ```
   db2trc off
   ```
6. Format the data as an ASCII file. You can sort the data two ways:
   - Use the **flw** option to sort the data by process or thread. For example:

     ```
     db2trc flw january23trace.dmp january23trace.flw
     ```
   - Use the **fmt** option to list every event chronologically. For example:

     ```
     db2trc fmt january23trace.dmp january23trace.fmt
     ```

DB2 XML Extender

Starting the trace for XML Extender

Purpose:

Records the XML Extender server activity. To start the trace, apply the `on` option to **dxxtrc**, along with the user profile and the name of an existing directory to contain the trace file. When the trace is turned on, the file, `dxx`*`INSTANCE`*`.trc`, is placed in the specified directory. *INSTANCE* is the value of DB2INSTANCE. Each DB2 UDB instance has its own log file. *INSTANCE* is the numeric UID value assigned to the User Profile for which the trace was started. The trace file is not limited in size.

Syntax:

Starting the trace from the Qshell:

```
►►—dxxtrc—on—user_profile—trace_directory——————————————————————————►◄
```

Starting the trace from the iSeries Navigator:

```
call schema.QZXMTRC('on', 'user_profile', 'trace_directory');
```

Starting the trace from the OS command line:

```
call QDBXM/QZXMTRC PARM(on user_profile 'trace_directory')
```

Parameters:

Table 6. Trace parameters

Parameter	Description
user_profile	The name of the user profile associated with the job within which the XML Extender is running.
trace_directory	Name of an existing path and directory where the `dxxINSTANCE.trc` is placed. Required, no default.

Examples:

The following examples show starting the trace, with file, `dxxdb2inst1.trc`, in the `/u/user1/dxx/trace` directory.

From the Qshell:

```
dxxtrc on user1 /u/user1/trace
```

From the iSeries Navigator:

```
call myschema.QZXMTRC('on', 'user1', '/u/user1/trace');
```

From the OS command line:

```
call QDBXM/QZXMTRC PARM(on user1 '/u/user1/trace')
```

Stopping the trace

Purpose:

Turns the trace off. Trace information is no longer logged.

Recommendation: Because running the trace log file size is not limited and can impact performance, turn trace off in a production environment.

Syntax:

Stopping the trace from the Qshell:

```
►►──dxxtrc──off──user_profile──────────────────────────────────────►◄
```

Stopping the trace from the iSeries Navigator:

```
call schema.QZXMTRC('off', 'user_profile');
```

Stopping the trace from the OS command line:

```
call QDBXM/QZXMTRC PARM(off user_profile)
```

Parameters:

Table 7. Trace parameters

Parameter	Description
user_profile	The name of the user profile associated with the job within which the XML Extender is running.

Examples:

The following examples demonstrate stopping the trace.

From the Qshell:

```
dxxtrc off user1
```

From the iSeries Navigator:

```
call myschema.QZXMTRC('off', 'user1');
```

From the OS command line:

```
call QDBXM/QZXMTRC PARM(off user1)
```

XML Extender UDF return codes

Embedded SQL statements return codes in the SQLCODE, SQLWARN, and SQLSTATE fields of the SQLCA structure. This structure is defined in an SQLCA INCLUDE file. (For more information about the SQLCA structure and SQLCA INCLUDE file, see the *DB2 Application Development Guide*.)

DB2 CLI calls return SQLCODE and SQLSTATE values that you can retrieve using the SQLError function. (For more information about retrieving error information with the SQLError function, see the *CLI Guide and Reference*.)

An SQLCODE value of 0 means that the statement ran successfully (with possible warning conditions). A positive SQLCODE value means that the statement ran successfully but with a warning. (Embedded SQL statements return information about the warning that is associated with 0 or positive SQLCODE values in the SQLWARN field.) A negative SQLCODE value means that an error occurred.

DB2 associates a message with each SQLCODE value. If an XML Extender UDF encounters a warning or error condition, it passes associated information to DB2 UDB for inclusion in the SQLCODE message.

Embedded SQL statements and DB2 UDB CLI calls that invoke the DB2 XML Extender UDFs might return SQLCODE messages and SQLSTATE values that are unique to these UDFs, but DB2 UDB returns these values in the same way that it does for other embedded SQL statements or other DB2 UDB CLI calls. Thus, the way that you access these values is the same as for embedded SQL statements or DB2 UDB CLI calls that do not start the DB2 UDB XML Extender UDFs.

XML Extender stored procedure return codes

The XML Extender provides return codes to help resolve problems with stored procedures. When you receive a return code from a stored procedure, check the following file, which matches the return code with an XML Extender error message number and the symbolic constant.

```
dxx_install/include/dxxrc.h
```

SQLSTATE codes and associated message numbers for XML Extender

Table 8. SQLSTATE codes and associated message numbers

SQLSTATE	Message Number	Description
00000	DXX*nnnn*I	No error occurred.
01HX0	DXXD003W	The element or attribute specified in the path expression is missing from the XML document.
38X00	DXXC000E	The XML Extender is unable to open the specified file.

Table 8. SQLSTATE codes and associated message numbers (continued)

SQLSTATE	Message Number	Description
38X01	DXXA072E	XML Extender tried to automatically bind the database before enabling it, but could not find the bind files.
	DXXC001E	The XML Extender could not find the file specified.
38X02	DXXC002E	The XML Extender is unable to read data from the specified file.
38X03	DXXC003E	The XML Extender is unable to write data to the file.
	DXXC011E	The XML Extender is unable to write data to the trace control file.
38X04	DXXC004E	The XML Extender was unable to operate the specified locator.
38X05	DXXC005E	The file size is greater than the XMLVarchar size, and the XML Extender is unable to import all the data from the file.
38X06	DXXC006E	The file size is greater than the size of the XMLCLOB, and the XML Extender is unable to import all the data from the file.
38X07	DXXC007E	The number of bytes in the LOB locator does not equal the file size.
38X08	DXXD001E	A scalar extraction function used a location path that occurs multiple times. A scalar function can use only a location path that does not have multiple occurrence.
38X09	DXXD002E	The path expression is syntactically incorrect.
38X10	DXXG002E	The XML Extender was unable to allocate memory from the operating system.
38X11	DXXA009E	This stored procedure is for an XML column only.
38X12	DXXA010E	While attempting to enable the column, the XML Extender could not find the DTD ID, which is the identifier specified for the DTD in the document access definition (DAD) file.
	DXXQ060E	The XML Extender could not find the SCHEMA ID while attempting to enable the column. The SCHEMA ID corresponds to the value of the location attribute of the nonamespacelocation tag which is inside the schemabindings tag in the DAD file.

Table 8. SQLSTATE codes and associated message numbers (continued)

SQLSTATE	Message Number	Description
38X14	DXXD000E	There was an attempt to store an invalid document into a table. Validation failed.
38X15	DXXA056E	The validation element in the document access definition (DAD) file is wrong or missing.
	DXXA057E	The name attribute of a side table in the document access definition (DAD) file is wrong or missing.
	DXXA058E	The name attribute of a column in the document access definition (DAD) file is wrong or missing.
	DXXA059E	The type attribute of a column in the document access definition (DAD) file is wrong or missing.
	DXXA060E	The path attribute of a column in the document access definition (DAD) file is wrong or missing.
	DXXA061E	The multi_occurrence attribute of a column in the document access definition (DAD) file is wrong or missing.
	DXXQ000E	A mandatory element is missing from the document access definition (DAD) file.
	DXXQ056E	The specified element/attribute cannot be mapped to a column that is specified as part of a foreign key. Data values for foreign keys are determined by that of the primary keys; a mapping of the specified element/attribute in the XML document to a table and column is not necessary.
	DXXQ057E	The schemabindings and DTD ID tags cannot exist together in the DAD file.
	DXXQ058E	The nonamespacelocation tag inside the schemabindings tag is missing in the DAD file.
	DXXQ059E	The doctype tag cannot be located inside the XCollection tag in the DAD for schema validation.
	DXXQ062E	This error condition is usually caused by a missing multi_occurrence = YES specification on the parent element_node of the given element or attribute.

Table 8. SQLSTATE codes and associated message numbers (continued)

SQLSTATE	Message Number	Description
	DXXQ063E	The value of the multi_occurrence attribute on the specified element_node in the document access definition (DAD) file is wrong or missing. The value must be 'yes' or 'no', case insensitive.
	DXXQ064E	A key column specified in the join condition was not mapped to any element or attribute node.
38X16	DXXG004E	A null value for a required parameter was passed to an XML stored procedure.
38X17	DXXQ001E	The SQL statement in the document access definition (DAD) file or the statement that overrides it is not valid. A SELECT statement is required for generating XML documents.
38X18	DXXG001E	XML Extender encountered an internal error.
	DXXG006E	XML Extender encountered an internal error while using CLI.
38X19	DXXQ002E	The system is running out of space in memory or disk. There is no space to contain the resulting XML documents.
38X20	DXXQ003W	The user-defined SQL query generates more XML documents than the specified maximum. Only the specified number of documents are returned.
38X21	DXXQ004E	The specified column is not one of the columns in the result of the SQL query.
38X22	DXXQ005E	The mapping of the SQL query to XML is incorrect.
38X23	DXXQ006E	An attribute_node element in the document access definition (DAD) file does not have a name attribute.
38X24	DXXQ007E	The attribute_node element in the document access definition (DAD) does not have a column element or RDB_node.
38X25	DXXQ008E	A text_node element in the document access definition (DAD) file does not have a column element.
38X26	DXXQ009E	The specified result table could not be found in the system catalog.

Table 8. SQLSTATE codes and associated message numbers (continued)

SQLSTATE	Message Number	Description
38X27	DXXQ010E DXXQ040E	The RDB_node of the attribute_node or text_node must have a table.
	DXXQ011E	The RDB_node of the attribute_node or text_node must have a column.
	DXXQ017E	An XML document generated by the XML Extender is too large to fit into the column of the result table.
	DXXQ040E	The specified element name in document access definition (DAD) file is wrong.
38X28	DXXQ012E	XML Extender could not find the expected element while processing the DAD.
	DXXQ016E	All tables must be defined in the RDB_node of the top element in the document access definition (DAD) file. Sub-element tables must match the tables defined in the top element. The table name in this RDB_node is not in the top element.
38X29	DXXQ013E	The element table or column must have a name in the document access definition (DAD) file.
	DXXQ015E	The condition in the condition element in the document access definition (DAD) file has an invalid format.
	DXXQ061E	The format of the string representation is invalid. If the string is a date, time, or timestamp value, the syntax does not conform to its data type.
38X30	DXXQ014E	An element_node element in the document access definition (DAD) file does not have a name attribute.
	DXXQ018E	The ORDER BY clause is missing from the SQL statement in a document access definition (DAD) file that maps SQL to XML.
38X31	DXXQ019E	The objids element does not have a column element in the document access definition (DAD) file that maps SQL to XML.
38x33	DXXG005E	This parameter is not supported in this release. It will be supported in the future release.
38x34	DXXG000E	An invalid file name was specified.
38X36	DXXA073E	The database was not bound when you tried to enable it.

Table 8. SQLSTATE codes and associated message numbers (continued)

SQLSTATE	Message Number	Description
38X37	DXXG007E	The server operating system locale is inconsistent with the DB2 UDB code page.
38X38	DXXG008E	The server operating system locale can not be found in the code page table.
38X41	DXXQ048E	The stylesheet processor returned an internal error. The XML document or the stylesheet might not be vaild.
38X42	DXXQ049E	The specified output file already exists in this directory.
38X43	DXXQ050E	The UDF was unable to create a unique file name for the output document in the specified directory because it does not have access. All file names that can be generated are in use or directory might not exist.
38X44	DXXQ051E	One or more input or output parameters have no valid value.
38X45	DXXQ055E	ICU error encountered during conversion operation.

XML Extender messages

DXXA000I Enabling column <*column_name*>. Please Wait.

Explanation: This is an informational message.

User Response: No action required.

DXXA001S An unexpected error occurred in build <*build_ID*>, file <*file_name*>, and line <*line_number*>.

Explanation: An unexpected error occurred.

User Response: If this error persists, contact your Software Service Provider. When reporting the error, be sure to include all the message text, the trace file, and an explanation of how to reproduce the problem.

DXXA002I Connecting to database <*database*>.

Explanation: This is an informational message.

User Response: No action required.

DXXA003E Cannot connect to database <*database*>.

Explanation: The database specified might not exist or could be corrupted.

User Response:

1. Ensure the database is specified correctly.
2. Ensure the database exists and is accessible.
3. Determine if the database is corrupted. If it is, ask your database administrator to recover it from a backup.

DXXA004E Cannot enable database <*database*>.

Explanation: The database might already be enabled or might be corrupted.

User Response:

1. Determine if the database is enabled.
2. Determine if the database is corrupted. If it is, ask your database administrator to recover it from a backup.

DXXA005I Enabling database <*database*>. Please wait.

Explanation: This is an informational message.

User Response: No action required.

DXXA006I The database <*database*> was enabled successfully.

Explanation: This is an informational message.

User Response: No action required.

DXXA007E Cannot disable database *<database>***.**

Explanation: The database cannot be disabled by XML Extender if it contains any XML columns or collections.

User Response: Backup any important data, disable any XML columns or collections, and update or drop any tables until there are no XML data types left in the database.

DXXA008I Disabling column *<column_name>***. Please Wait.**

Explanation: This is an information message.

User Response: No action required.

DXXA009E Xcolumn tag is not specified in the DAD file.

Explanation: This stored procedure is for XML Column only.

User Response: Ensure the Xcolumn tag is specified correctly in the DAD file.

DXXA010E Attempt to find DTD ID *<dtdid>* **failed.**

Explanation: While attempting to enable the column, the XML Extender could not find the DTD ID, which is the identifier specified for the DTD in the document access definition (DAD) file.

User Response: Ensure the correct value for the DTD ID is specified in the DAD file.

DXXA011E Inserting a record into DB2XML.XML_USAGE table failed.

Explanation: While attempting to enable the column, the XML Extender could not insert a record into the DB2XML.XML_USAGE table.

User Response: Ensure the DB2XML.XML_USAGE table exists and that a record by the same name does not already exist in the table.

DXXA012E Attempt to update DB2XML.DTD_REF table failed.

Explanation: While attempting to enable the column, the XML Extender could not update the DB2XML.DTD_REF table.

User Response: Ensure the DB2XML.DTD_REF table exists. Determine whether the table is corrupted or if the administration user ID has the correct authority to update the table.

DXXA013E Attempt to alter table *<table_name>* **failed.**

Explanation: While attempting to enable the column, the XML Extender could not alter the specified table.

User Response: Check the privileges required to alter the table.

DXXA014E The specified root ID column: *<root_id>* **is not a single primary key of table** *<table_name>***.**

Explanation: The root ID specified is either not a key, or it is not a single key of table *table_name*.

User Response: Ensure the specified root ID is the single primary key of the table.

DXXA015E The column DXXROOT_ID already exists in table *<table_name>***.**

Explanation: The column DXXROOT_ID exists, but was not created by XML Extender.

User Response: Specify a primary column for the root ID option when enabling a column, using a different different column name.

DXXA016E The input table *<table_name>* **does not exist.**

Explanation: The XML Extender was unable to find the specified table in the system catalog.

User Response: Ensure that the table exists in the database, and is specified correctly.

DXXA017E The input column *<column_name>* **does not exist in the specified table** *<table_name>***.**

Explanation: The XML Extender was unable to find the column in the system catalog.

User Response: Ensure the column exists in a user table.

DXXA018E The specified column is not enabled for XML data.

Explanation: While attempting to disable the column, XML Extender could not find the column in the DB2XML.XML_USAGE table, indicating that the column is not enabled. If the column is not XML-enabled, you do not need to disable it.

User Response: No action required.

DXXA019E A input parameter required to enable the column is null.

Explanation: A required input parameter for the enable_column() stored procedure is null.

User Response: Check all the input parameters for the enable_column() stored procedure.

DXXA020E Columns cannot be found in the table *<table_name>***.**

Explanation: While attempting to create the default view, the XML Extender could not find columns in the specified table.

User Response: Ensure the column and table name are specified correctly.

DXXA021E Cannot create the default view *<default_view>***.**

Explanation: While attempting to enable a column, the XML Extender could not create the specified view.

User Response: Ensure that the default view name is unique. If a view with the name already exists, specify a unique name for the default view.

DXXA022I Column *<column_name>* **enabled.**

Explanation: This is an informational message.

User Response: No response required.

DXXA023E Cannot find the DAD file.

Explanation: While attempting to disable a column, the XML Extender was unable to find the document access definition (DAD) file.

User Response: Ensure you specified the correct database name, table name, or column name.

DXXA024E The XML Extender encountered an internal error while accessing the system catalog tables.

Explanation: The XML Extender was unable to access system catalog table.

User Response: Ensure the database is in a stable state.

DXXA025E Cannot drop the default view *<default_view>***.**

Explanation: While attempting to disable a column, the XML Extender could not drop the default view.

User Response: Ensure the administration user ID for XML Extender has the privileges necessary to drop the default view.

DXXA026E Unable to drop the side table *<side_table>***.**

Explanation: While attempting to disable a column, the XML Extender was unable to drop the specified table.

User Response: Ensure that the administrator user ID for XML Extender has the privileges necessary to drop the table.

DXXA027E Could not disable the column.

Explanation: XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA028E Could not disable the column.

Explanation: XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA029E Could not disable the column.

Explanation: XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA030E Could not disable the column.

Explanation: XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA031E Unable to reset the DXXROOT_ID column value in the application table to NULL.

Explanation: While attempting to disable a column, the XML Extender was unable to set the value of DXXROOT_ID in the application table to NULL.

User Response: Ensure that the administrator user ID for XML Extender has the privileges necessary to alter the application table.

DXXA032E Decrement of USAGE_COUNT in DB2XML.XML_USAGE table failed.

Explanation: While attempting to disable the column, the XML Extender was unable to reduce the value of the USAGE_COUNT column by one.

User Response: Ensure that the DB2XML.XML_USAGE table exists and that the administrator user ID for XML Extender has the necessary privileges to update the table.

DXXA033E Attempt to delete a row from the DB2XML.XML_USAGE table failed.

Explanation: While attempting to disable a column, the XML Extender was unable to delete its associate row in the DB2XML.XML_USAGE table.

User Response: Ensure that the DB2XML.XML_USAGE table exists and that the administration user ID for XML Extender has the privileges necessary to update this table.

DXXA034I XML Extender has successfully disabled column *<column_name>***.**

Explanation: This is an informational message

User Response: No action required.

DXXA035I XML Extender is disabling database *<database>***. Please wait.**

Explanation: This is an informational message.

User Response: No action is required.

DXXA036I XML Extender has successfully disabled database *<database>***.**

Explanation: This is an informational message.

User Response: No action is required.

DXXA037E The specified table space name is longer than 18 characters.

Explanation: The table space name cannot be longer than 18 alphanumeric characters.

User Response: Specify a name less than 18 characters.

DXXA038E The specified default view name is longer than 18 characters.

Explanation: The default view name cannot be longer than 18 alphanumeric characters.

User Response: Specify a name less than 18 characters.

DXXA039E The specified ROOT_ID name is longer than 18 characters.

Explanation: The ROOT_ID name cannot be longer than 18 alphanumeric characters.

User Response: Specify a name less than 18 characters.

DXXA046E Unable to create the side table *<side_table>***.**

Explanation: While attempting to enable a column, the XML Extender was unable to create the specified side table.

User Response: Ensure that the administrator user ID for XML Extender has the privileges necessary to create the side table.

DXXA047E Could not enable the column.

Explanation: XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA048E Could not enable the column.

Explanation: XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA049E Could not enable the column.

Explanation: XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA050E Could not enable the column.

Explanation: XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA051E Could not disable the column.

Explanation: XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The system is out of memory.
- A trigger with this name does not exist.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA052E Could not disable the column.

Explanation: XML Extender could not disable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA053E Could not enable the column.

Explanation: XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA054E Could not enable the column.

Explanation: XML Extender could not enable a column because an internal trigger failed. Possible causes:

- The DAD file has incorrect syntax.
- The system is out of memory.
- Another trigger exists with the same name.

User Response: Use the trace facility to create a trace file and try to correct the problem. If the problem persists, contact your Software Service Provider and provide the trace file.

DXXA056E The validation value <*validation_value*> **in the DAD file is invalid.**

Explanation: The validation element in document access definition (DAD) file is wrong or missing.

User Response: Ensure that the validation element is specified correctly in the DAD file.

DXXA057E A side table name <*side_table_name*> **in DAD is invalid.**

Explanation: The name attribute of a side table in the document access definition (DAD) file is wrong or missing.

User Response: Ensure that the name attribute of a side table is specified correctly in the DAD file.

DXXA058E A column name <*column_name*> **in the DAD file is invalid.**

Explanation: The name attribute of a column in the document access definition (DAD) file is wrong or missing.

User Response: Ensure that the name attribute of a column is specified correctly in the DAD file.

DXXA059E The type <*column_type*> of column <*column_name*> in the DAD file is invalid.

Explanation: The type attribute of a column in the document access definition (DAD) file is wrong or missing.

User Response: Ensure that the type attribute of a column is specified correctly in the DAD file.

DXXA060E The path attribute <*location_path*> of <*column_name*> in the DAD file is invalid.

Explanation: The path attribute of a column in the document access definition (DAD) file is wrong or missing.

User Response: Ensure that the path attribute of a column is specified correctly in the DAD file.

DXXA061E The multi_occurrence attribute <*multi_occurrence*> of <*column_name*> in the DAD file is invalid.

Explanation: The multi_occurrence attribute of a column in the document access definition (DAD) file is wrong or missing.

User Response: Ensure that the multi_occurrence attribute of a column is specified correctly in the DAD file.

DXXA062E Unable to retrieve the column number for <*column_name*> in table <*table_name*>.

Explanation: XML Extender could not retrieve the column number for *column_name* in table *table_name* from the system catalog.

User Response: Make sure the application table is well defined.

DXXA063I Enabling collection <*collection_name*>. Please Wait.

Explanation: This is an information message.

User Response: No action required.

DXXA064I Disabling collection <*collection_name*>. Please Wait.

Explanation: This is an information message.

User Response: No action required.

DXXA065E Calling stored procedure <*procedure_name*> failed.

Explanation: Check the shared library db2xml and see if the permission is correct.

User Response: Make sure the client has permission to run the stored procedure.

DXXA066I XML Extender has successfully disabled collection <*collection_name*>.

Explanation: This is an informational message.

User Response: No response required.

DXXA067I XML Extender has successfully enabled collection <*collection_name*>.

Explanation: This is an informational message.

User Response: No response required.

DXXA068I XML Extender has successfully turned the trace on.

Explanation: This is an informational message.

User Response: No response required.

DXXA069I XML Extender has successfully turned the trace off.

Explanation: This is an informational message.

User Response: No response required.

DXXA070W The database has already been enabled.

Explanation: The enable database command was executed on the enabled database

User Response: No action is required.

DXXA071W The database has already been disabled.

Explanation: The disable database command was executed on the disabled database

User Response: No action is required.

DXXA072E XML Extender couldn't find the bind files. Bind the database before enabling it.

Explanation: XML Extender tried to automatically bind the database before enabling it, but could not find the bind files

User Response: Bind the database before enabling it.

DXXA073E The database is not bound. Please bind the database before enabling it.

Explanation: The database was not bound when user tried to enable it.

User Response: Bind the database before enabling it.

DXXA074E Wrong parameter type. The stored procedure expects a STRING parameter.

Explanation: The stored procedure expects a STRING parameter.

User Response: Declare the input parameter to be STRING type.

DXXA075E Wrong parameter type. The input parameter should be a LONG type.

Explanation: The stored procedure expects the input parameter to be a LONG type.

User Response: Declare the input parameter to be a LONG type.

DXXA076E XML Extender trace instance ID invalid.

Explanation: Cannot start trace with the instance ID provided.

User Response: Ensure that the instance ID is a valid iSeries user ID.

DXXA077E The license key is not valid. See the server error log for more detail.

Explanation: The software license has expired or does not exist.

User Response: Contact your service provider to obtain a new software license.

DXXC000E Unable to open the specified file.

Explanation: The XML Extender is unable to open the specified file.

User Response: Ensure that the application user ID has read and write permission for the file.

DXXC001E The specified file is not found.

Explanation: The XML Extender could not find the file specified.

User Response: Ensure that the file exists and the path is specified correctly.

DXXC002E Unable to read file.

Explanation: The XML Extender is unable to read data from the specified file.

User Response: Ensure that the application user ID has read permission for the file.

DXXC003E Unable to write to the specified file.

Explanation: The XML Extender is unable to write data to the file.

User Response: Ensure that the application user ID has write permission for the file or that the file system has sufficient space.

DXXC004E Unable to operate the LOB Locator: rc=<*locator_rc*>.

Explanation: The XML Extender was unable to operate the specified locator.

User Response: Ensure the LOB Locator is set correctly.

DXXC005E Input file size is greater than XMLVarchar size.

Explanation: The file size is greater than the XMLVarchar size and the XML Extender is unable to import all the data from the file.

User Response: Use the XMLCLOB column type.

DXXC006E The input file exceeds the DB2 UDB LOB limit.

Explanation: The file size is greater than the size of the XMLCLOB and the XML Extender is unable to import all the data from the file.

User Response: Decompose the file into smaller objects or use an XML collection.

DXXC007E Unable to retrieve data from the file to the LOB Locator.

Explanation: The number of bytes in the LOB Locator does not equal the file size.

User Response: Ensure the LOB Locator is set correctly.

DXXC008E Can not remove the file <*file_name*>.

Explanation: The file has a sharing access violation or is still open.

User Response: Close the file or stop any processes that are holding the file. You might have to stop and restart DB2.

DXXC009E Unable to create file to <*directory*> directory.

Explanation: The XML Extender is unable to create a file in directory *directory*.

User Response: Ensure that the directory exists, that the application user ID has write permission for the directory, and that the file system has sufficient space for the file.

DXXC010E Error while writing to file <*file_name*>.

Explanation: There was an error while writing to the file *file_name*.

User Response: Ensure that the file system has sufficient space for the file.

DXXC011E Unable to write to the trace control file.

Explanation: The XML Extender is unable to write data to the trace control file.

User Response: Ensure that the application user ID has write permission for the file or that the file system has sufficient space.

DXXC012E Cannot create temporary file.

Explanation: Cannot create file in system temp directory.

User Response: Ensure that the application user ID has write permission for the file system temp directory or that the file system has sufficient space for the file.

DXXC013E The results of the extract UDF exceed the size limit for the UDF return type.

Explanation: The data returned by an extract UDF must fit into the size limit of the return type of the UDF, as defined in the DB2 UDB XML Extender Administration and Programming guide. For example, the results of extractVarchar must be no more than 4000 bytes (including the terminating NULL).

User Response: Use an extract UDF that has a larger size limit for the return type: 254 bytes for extractChar(), 4 KB for extractVarchar(), and 2 GB for extractClob().

DXXD000E An invalid XML document is rejected.

Explanation: There was an attempt to store an invalid document into a table. Validation has failed.

User Response: Check the document with its DTD using an editor that can view invisible invalid characters. To suppress this error, turn off validation in the DAD file.

DXXD001E <*location_path*> occurs multiple times.

Explanation: A scalar extraction function used a location path that occurs multiple times. A scalar function can only use a location path that does not have multiple occurrences.

User Response: Use a table function (add an 's' to the end of the scalar function name).

DXXD002E A syntax error occurred near position <*position*> in the search path.

Explanation: The path expression is syntactically incorrect.

User Response: Correct the search path argument of the query. Refer to the documentation for the syntax of path expressions.

DXXD003W Path not found. Null is returned.

Explanation: The element or attribute specified in the path expression is missing from the XML document.

User Response: Verify that the specified path is correct.

DXXG000E The file name <*file_name*> is invalid.

Explanation: An invalid file name was specified.

User Response: Specify a correct file name and try again.

DXXG001E An internal error occurred in build <*build_ID*>, file <*file_name*>, and line <*line_number*>.

Explanation: XML Extender encountered an internal error.

User Response: Contact your Software Service Provider. When reporting the error, be sure to include all the messages, the trace file and how to reproduce the error.

DXXG002E The system is out of memory.

Explanation: The XML Extender was unable to allocate memory from the operating system.

User Response: Close some applications and try again. If the problem persists, refer to your operating system documentation for assistance. Some operating systems might require that you reboot the system to correct the problem.

DXXG004E Invalid null parameter.

Explanation: A null value for a required parameter was passed to an XML stored procedure.

User Response: Check all required parameters in the argument list for the stored procedure call.

DXXG005E Parameter not supported.

Explanation: This parameter is not supported in this release, will be supported in the future release.

User Response: Set this parameter to NULL.

DXXG006E Internal Error CLISTATE=<*clistate*>, RC=<*cli_rc*>, build <*build_ID*>, file <*file_name*>, line <*line_number*> CLIMSG=<*CLI_msg*>.

Explanation: XML Extender encountered an internal error while using CLI.

User Response: Contact your Software Service Provider. Potentially this error can be caused by incorrect user input. When reporting the error, be sure to include all output messages, trace log, and how to reproduce the problem. Where possible, send any DADs, XML documents, and table definitions which apply.

DXXG007E Locale <*locale*> is inconsistent with DB2 UDB code page <*code_page*>.

Explanation: The server operating system locale is inconsistent with DB2 UDB code page.

User Response: Correct the server operating system locale and restart DB2.

DXXG008E Locale <*locale*> is not supported.

Explanation: The server operating system locale can not be found in the code page table.

User Response: Correct the server operating system locale and restart DB2.

DXXG017E The limit for *XML_Extender_constant* has been exceeded in build *build_ID*, file *file_name*, and line *line_number*.

Explanation: Check the XML Extender Administration and Programming Guide to see whether your application has exceeded a value in the limits table. If no limit has been exceeded, contact your Software Service Provider. When reporting the error, include all output messages, trace files, and information on how to reproduce the problem such as input DADs, XML documents, and table definitions.

User Response: Correct the server operating system locale and restart DB2.

DXXM001W A DB2 UDB error occurred.

Explanation: DB2 encountered the specified error.

User Response: See any accompanying messages for futher explanation and refer to DB2 UDB messages and codes documentation for your operating system.

DXXQ000E <*Element*> is missing from the DAD file.

Explanation: A mandatory element is missing from the document access definition (DAD) file.

User Response: Add the missing element to the DAD file.

DXXQ001E Invalid SQL statement for XML generation.

Explanation: The SQL statement in the document access definition (DAD) or the one that overrides it is not valid. A SELECT statement is required for generating XML documents.

User Response: Correct the SQL statement.

DXXQ002E Cannot generate storage space to hold XML documents.

Explanation: The system is running out of space in memory or disk. There is no space to contain the resulting XML documents.

User Response: Limit the number of documents to be generated. Reduce the size of each documents by removing some unnecessary element and attribute nodes from the document access definition (DAD) file.

DXXQ003W Result exceeds maximum.

Explanation: The user-defined SQL query generates more XML documents than the specified maximum. Only the specified number of documents are returned.

User Response: No action is required. If all documents are needed, specify zero as the maximum number of documents.

DXXQ004E The column <*column_name*> is not in the result of the query.

Explanation: The specified column is not one of the columns in the result of the SQL query.

User Response: Change the specified column name in the document access definition (DAD) file to make it one of the columns in the result of the SQL query. Alternatively, change the SQL query so that it has the specified column in its result.

DXXQ005E Wrong relational mapping. The element <*element_name*> is at a lower level than its child column <*column_name*>.

Explanation: The mapping of the SQL query to XML is incorrect.

User Response: Make sure that the columns in the result of the SQL query are in a top-down order of the relational hierarchy. Also make sure that there is a single-column candidate key to begin each level. If such a key is not available in a table, the query should generate one for that table using a table expression and the DB2 UDB built-in function generate_unique().

DXXQ006E An attribute_node element has no name.

Explanation: An attribute_node element in the document access definition (DAD) file does not have a name attribute.

User Response: Ensure that every attribute_node has a name in the DAD file.

DXXQ007E The attribute_node <*attribute_name*> has no column element or RDB_node.

Explanation: The attribute_node element in the document access definition (DAD) does not have a column element or RDB_node.

User Response: Ensure that every attribute_node has a column element or RDB_node in the DAD.

DXXQ008E A text_node element has no column element.

Explanation: A text_node element in the document access definition (DAD) file does not have a column element.

User Response: Ensure that every text_node has a column element in the DAD.

DXXQ009E Result table <*table_name*> does not exist.

Explanation: The specified result table could not be found in the system catalog.

User Response: Create the result table before calling the stored procedure.

DXXQ010E RDB_node of <*node_name*> does not have a table in the DAD file.

Explanation: The RDB_node of the attribute_node or text_node must have a table.

User Response: Specify the table of RDB_node for attribute_node or text_node in the document access definition (DAD) file.

DXXQ011E RDB_node element of <*node_name*> does not have a column in the DAD file.

Explanation: The RDB_node of the attribute_node or text_node must have a column.

User Response: Specify the column of RDB_node for attribute_node or text_node in the document access definition (DAD) file.

DXXQ012E Errors occurred in DAD.

Explanation: XML Extender could not find the expected element while processing the DAD.

User Response: Check that the DAD is a valid XML document and contains all the elements required by the DAD DTD. Consult the XML Extender publication for the DAD DTD.

DXXQ013E The table or column element does not have a name in the DAD file.

Explanation: The element table or column must have a name in the document access definition (DAD) file.

User Response: Specify the name of table or column element in the DAD.

DXXQ014E An element_node element has no name.

Explanation: An element_node element in the document access definition (DAD) file does not have a name attribute.

User Response: Ensure that every element_node element has a name in the DAD file.

DXXQ015E The condition format is invalid.

Explanation: The condition in the condition element in the document access definition (DAD) has an invalid format.

User Response: Ensure that the format of the condition is valid.

DXXQ016E The table name in this RDB_node is not defined in the top element of the DAD file.

Explanation: All tables must be defined in the RDB_node of the top element in the document access definition (DAD) file. Sub-element tables must match the tables defined in the top element. The table name in this RDB_node is not in the top element.

User Response: Ensure that the table of the RDB node is defined in the top element of the DAD file.

DXXQ017E The column in the result table *<table_name>* **is too small.**

Explanation: An XML document generated by the XML Extender is too large to fit into the column of the result table.

User Response: Drop the result table. Create another result table with a bigger column. Rerun the stored procedure.

DXXQ018E The ORDER BY clause is missing from the SQL statement.

Explanation: The ORDER BY clause is missing from the SQL statement in a document access definition (DAD) file that maps SQL to XML.

User Response: Edit the DAD file. Add an ORDER BY clause that contains the entity-identifying columns.

DXXQ019E The element objids has no column element in the DAD file.

Explanation: The objids element does not have a column element in the document access definition (DAD) file that maps SQL to XML.

User Response: Edit the DAD file. Add the key columns as sub-elements of the element objids.

DXXQ020I XML successfully generated.

Explanation: The requested XML documents have been successfully generated from the database.

User Response: No action is required.

DXXQ021E Table *<table_name>* **does not have column** *<column_name>***.**

Explanation: The table does not have the specified column in the database.

User Response: Specify another column name in DAD or add the specified column into the table database.

DXXQ022E Column *<column_name>* **of** *<table_name>* **should have type** *<type_name>***.**

Explanation: The type of the column is wrong.

User Response: Correct the type of the column in the document access definition (DAD).

DXXQ023E Column *<column_name>* **of** *<table_name>* **cannot be longer than** *<length>***.**

Explanation: The length defined for the column in the DAD is too long.

User Response: Correct the column length in the document access definition (DAD).

DXXQ024E Can not create table *<table_name>***.**

Explanation: The specified table can not be created.

User Response: Ensure that the user ID creating the table has the necessary authority to create a table in the database.

DXXQ025I XML decomposed successfully.

Explanation: An XML document has been decomposed and stored in a collection successfully.

User Response: No action is required.

DXXQ026E XML data *<xml_name>* **is too large to fit in column** *<column_name>***.**

Explanation: The specified piece of data from an XML document is too large to fit into the specified column.

User Response: Increase the length of the column using the ALTER TABLE statement or reduce the size of the data by editing the XML document.

DXXQ028E Cannot find the collection *<collection_name>* **in the XML_USAGE table.**

Explanation: A record for the collection cannot be found in the XML_USAGE table.

User Response: Verify that you have enabled the collection.

DXXQ029E Cannot find the DAD in XML_USAGE table for the collection *<collection_name>***.**

Explanation: A DAD record for the collection cannot be found in the XML_USAGE table.

User Response: Ensure that you have enabled the collection correctly.

DXXQ030E Wrong XML override syntax.

Explanation: The XML_override value is specified incorrectly in the stored procedure.

User Response: Ensure that the syntax of XML_override is correct.

DXXQ031E Table name cannot be longer than maximum length allowed by DB2.

Explanation: The table name specified by the condition element in the DAD is too long.

User Response: Correct the length of the table name in document access definition (DAD).

DXXQ032E Column name cannot be longer than maximum length allowed by DB2.

Explanation: The column name specified by the condition element in the DAD is too long.

User Response: Correct the length of the column name in the document access definition (DAD).

DXXQ033E Invalid identifier starting at *<identifier>*

Explanation: The string is not a valid DB2 UDB SQL identifier.

User Response: Correct the string in the DAD to conform to the rules for DB2 UDB SQL identifiers.

DXXQ034E Invalid condition element in top RDB_node of DAD: *<condition>*

Explanation: The condition element must be a valid WHERE clause consisting of join conditions connected by the conjunction AND.

User Response: See the XML Extender documentation for the correct syntax of the join condition in a DAD.

DXXQ035E Invalid join condition in top RDB_node of DAD: *<condition>*

Explanation: Column names in the condition element of the top RDB_node must be qualified with the table name if the DAD specifies multiple tables.

User Response: See the XML Extender documentation for the correct syntax of the join condition in a DAD.

DXXQ036E A Schema name specified under a DAD condition tag is longer than allowed.

Explanation: An error was detected while parsing text under a condition tag within the DAD. The condition text contains an id qualified by a schema name that is too long.

User Response: Correct the text of the condition tags in document access definition (DAD).

DXXQ037E Cannot generate *<element>* **with multiple occurrences.**

Explanation: The element node and its descendents have no mapping to database, but its multi_occurrence equals YES.

User Response: Correct the DAD by either setting the multi_occurrence to NO or create a RDB_node in one of its descendents.

DXXQ038E The SQL statement is too long: SQL_statement

Explanation: The SQL statement specified in the *<SQL_stmt>* element of DAD exceeds the allowed number of bytes.

User Response: Reduce the length of the SQL statement to less than or equal to 32765 bytes for Windows and UNIX, or 16380 bytes for OS/390 and iSeries.

DXXQ039E Too many columns specified for a table in the DAD file.

Explanation: A DAD file used for decomposition or RDB composition can have a maximum of 100 text_node and attribute_node elements that specify unique columns within the same table.

User Response: Reduce the total number of text_node and attribute_node elements that refer to unique columns within the same table 100 or less.

DXXQ040E The element name *<element_name>* **in the DAD file is invalid.**

Explanation: The specified element name in the document access definition (DAD) file is wrong.

User Response: Ensure that the element name is typed correctly in the DAD file. See the DTD for the DAD file.

DXXQ041W XML document successfully generated. One or more override paths specified is invalid and ignored.

Explanation: Specify only one override path.

User Response: Ensure that the element name is typed correctly in the DAD file. See the DTD for the DAD file.

DXXQ043E Attribute *<attr_name>* **not found under element** *<elem_name>***.**

Explanation: The attribute *<attr_name>* was not present in element *<elem_name>* or one of its child elements.

User Response: Ensure the attribute appears in the XML document everywhere that the DAD requires it.

DXXQ044E Element *<elem_name>* **does not have an ancestor element***<ancestor>***.**

Explanation: According to the DAD, *<ancestor>* is an ancestor element of *<elem_name>* . In the XML document, one or more element *<elem_name>* does not have such an ancestor.

User Response: Ensure that the nesting of elements in the XML document conforms to what is specified in the corresponding DAD.

DXXQ045E Subtree under element <*elem_name*> contains multiple attributes named<*attrib_name*>.

Explanation: A subtree under <*elem_name*> in the XML document contains multiple instances of attribute<*attrib_name*> , which according to the DAD, is to be decomposed into the same row. Elements or attributes that are to be decomposed must have unique names.

User Response: Ensure that the element or attribute in the subtree has a unique name.

DXXQ046W The DTD ID was not found in the DAD.

Explanation: In the DAD, VALIDATION is set to YES, but the DTDID element is not specified. No validation check is performed.

User Response: No action is required. If validation is needed, specify the DTDID element in the DAD file.

DXXQ047E Parser error on line *linenumber* **column** *colnumber***:** *msg*

Explanation: The parser could not parse the document because of the reported error.

User Response: Correct the error in the document, consulting the XML specifications if necessary.

DXXQ048E Internal error - see trace file.

Explanation: The stylesheet processor returned an internal error. The XML document or the stylesheet might not vaild.

User Response: Ensure the XML document and the stylesheet are valid.

DXXQ049E The output file already exists.

Explanation: The specified output file already exists in this directory.

User Response: Change the output path or file name for the output document to a unique name or delete the existing file.

DXXQ050E Unable to create a unique file name.

Explanation: The UDF was unable to create a unique file name for the output document in the specified directory because it does not have access, all file names that can be generated are in use or directory might not exist.

User Response: Ensure that the UDF has access to the specified directory, change to a directory with available file names.

DXXQ051E No input or output data.

Explanation: One or more input or output parameters have no valid value.

User Response: Check the statement to see if required parameters are missing.

DXXQ052E An error occurred while accessing the DB2XML.XML_USAGE table.

Explanation: Either the database has not been enabled or the table DB2XML.XML_USAGE has been dropped.

User Response: Ensure that the database has been enabled and the table DB2XML.XML_USAGE is accessible.

DXXQ053E An SQL statement failed : *msg*

Explanation: An SQL statement generated during XML Extender processing failed to execute.

User Response: Examine the trace for more details. If you cannot correct the error condition, contact your softwaresService provider. When reporting the error, be sure to include all the messages, the trace file and how to reproduce the error.

DXXQ054E Invalid input parameter: *param*

Explanation: The specified input parameter to a stored procedure or UDF is invalid.

User Response: Check the signature of the relevant stored procedure or UDF, and ensure the actual input parameter is correct.

DXXQ055E ICU error: *uerror*

Explanation: ICU error encountered during conversion operation.

User Response: Report the error to your software service provider. Include trace file, error message, and instructions to reproduce the error.

DXXQ056E Element/attribute *xmlname* **cannot be mapped to the column designated as part of the foreign key (column** *column* **in table** *table***).**

Explanation: The specified element/attribute cannot be mapped to a column that is specified as part of a foreign key. Data values for foreign keys are determined by that of the primary keys; a mapping of the specified element/attribute in the xml document to a table and column is not necessary.

User Response: Remove the RDB_node mapping to the specified column and table in the DAD.

DXXQ057E The schemabindings and dtdid tags cannot exist together in the DAD file.

Explanation: The schemabindings and dtdid tags cannot exist together in the DAD file.

User Response: Check that either the schemabindings tag or the dtdid tag exists in the DAD file, but not both.

DXXQ058E The nonamespacelocation tag inside the schemabindings tag is missing in the DAD file.

Explanation: The nonamespacelocation tag inside the schemabindings tag is missing in the DAD file.

User Response: Add the nonamespacelocation tag to the schemabindings tag.

DXXQ059E The doctype tag cannot be located inside the XCollection tag in the DAD for schema validation.

Explanation: The doctype tag cannot be located inside the XCollection tag in the DAD for schema validation.

User Response: Remove the doctype tag inside the Xcollection tag for schema validation.

DXXQ060E Attempt to find SCHEMA ID *schemaid* failed.

Explanation: The XML Extender could not find the SCHEMA ID while attempting to enable the column. The SCHEMA ID corresponds to the value of the location attribute of the nonamespacelocation tag which is inside the schemabindings tag in the DAD file.

User Response: Check that the correct value for the SCHEMA ID is specified in the DAD file.

DXXQ061E The format of the string is invalid.

Explanation: The format of the string representation is invalid. If the string is a date, time, or timestamp value, the syntax does not conform to its data type.

User Response: Check that the format of the date, time, or timestamp value conforms to the format for its data type.

DXXQ062E No rows of result set for *table* are left to produce a XML value for *element*.

Explanation: This error condition is usually caused by a missing multi_occurrence = YES specification on the parent element_node of the given element or attribute.

User Response: Check the DAD that the value of multi_occurrence on the parent element_node correctly reflects the multiplicity of child element_nodes.

DXXQ063E The multi_occurrence attribute value on *elementname* in the DAD file is invalid.

Explanation: The value of the multi_occurrence attribute on the specified element_node in the document access definition (DAD) file is wrong or missing. The value must be 'yes' or 'no', case insensitive.

User Response: Ensure that the multi_occurrence attribute is specified correctly in the DAD file.

DXXQ064E Column *column* not found in foreign table *table*.

Explanation: A key column specified in the join condition was not mapped to any element or attribute node.

User Response: Check to make sure the join condition specified in the DAD file is correct, and all key columns are mapped to element or attribute nodes.

DXXQ065I All triggers relating to XML enabled columns have been successfully regenerated.

Explanation: This is an informational message only.

User Response: No action required.

DXXQ066E The primary key for table *tablename* does not exist.

Explanation: XML Extender could not determine the primary key for table *tablename*. Check that the primary key for the table was not dropped after the column was enabled for XML.

User Response: Alter the table to add the primary key specified as the ROOT ID when the column was enabled for XML.

DXXQ067E Attempt to *action* failed.

Explanation: While attempting to *action*, a SQL error occurred.

User Response: Contact your Software Service Provider. When reporting the error, be sure to include the XML Extender trace file.

DXXQ068E Cannot set current SQLID to [userid]. SQLCODE = [sqlcode].

Explanation: While attempting to set current sqlid to a secondary authorization id, a SQL error occurred.

User Response: Check that you are specifying a valid secondary authorization id and that you have authorization for the id.

Locking issues

This section will provide an overview of when locking issues may arise and how to investigate them. The main locking issues that you will encounter will be: lock waits, timeouts, escalations and deadlocks.

Lock wait and lock timeout:

An application that makes a request for a lock that is not compatible with the existing locks on the object, or a lock request not already satisfied will be placed into a lock request pending queue. The lock request will continue to be held for the waiting application until either timeout period is exceeded or a deadlock is the cause of the result.

The `LOCKTIMEOUT` database parameter, measured in seconds, is used to configure how long a period an application will wait for a lock to be made available. If the timeout period is exceeded, the waiting application receives an SQL0911 error message with a return code 68 and the application's unit of work is automatically rolled back by the database manager. The default value for `LOCKTIMEOUT` is -1, which causes lock timeouts to be disabled. That is, an application that is waiting for a lock will continue to wait indefinitely or until the lock is released. For transaction environments, the recommended starting value is 30 seconds, however, tuning may be necessary in order to find a more appropriate value for this field.

Lock escalation:

Lock escalation can occur in two different scenarios:

1. A single application requests a lock that will exceed its allowable number of locks.
2. An application triggers lock escalation because the maximum number of database locks on the system has been exceeded.

In both cases, the database manager will attempt to free up memory allocated to locking by obtaining table locks and releasing existing row locks. The desired effect is to make more lock memory available for additional applications.

The following two database parameters have a direct affect on lock escalation:

- locklist- the total number of 4k pages allocated for lock storage.
- maxlocks- the allowable percentage of locklist that can be used by a single application

Tuning and monitoring may be necessary to find a balance for these values. Workload, and query behavior dictate locking patterns and how applications will use lock memory.

Deadlocks:

Deadlocks occur when applications cannot complete a unit of work due to conflicting lock requests that cannot be resolved until the unit of work is completed. The DB2 deadlock detector is responsible for handling deadlocks. When a deadlock is detected, the deadlock detector will choose a victim that will be automatically rolled back and issued an SQL0911 R.C. 2. By rolling back the victim, the lock conflict is removed, and the other application can continue processing.

The frequency at which the DB2 deadlock detector checks for deadlocks can be controlled via DLCHKTIME(measured in milliseconds, from 1000 to 600000). Setting this value high will cause the deadlock check time to be increased but the overhead of deadlock detection will be removed. This could potentially result in applications stuck in deadlock for prolonged periods of time. Setting the deadlock check time to a smaller value allows for deadlocks to be detected sooner, however it also introduces additional overhead for the checking.

Related reference:

- Factors that affect locking
- Locks and performance
- Guidelines for locking
- Deadlocks between applications
- Specifying a lock wait mode strategy
- Correcting lock escalation problems
- Deadlock Prevention for Multiple Contexts

Tracking deadlocks

Snapshots provide valuable insight into the number of deadlocks that have occurred. If you find that the number of deadlocks is abnormally large, you should consider enabling an event monitor to track the deadlock events more closely. By default, all databases have an event monitor named `DB2DETAILDEADLOCK` defined, which keeps track of DEADLOCKS WITH DETAILS. The `DB2DETAILDEADLOCK` event monitor starts automatically when the database starts.

The following example demonstrates how a deadlock can be analyzed using an event monitor: Event monitor sample output.

Alternatively, you can analyze the deadlocks using the db2pd tool. Refer to **db2pd sample output**.

These examples shows how to identify both a deadlock event and the applications involved. You could potentially correct this issue by adjusting the lock mode or isolation level of the application. The *DB2 Administration Guide* is an excellent reference for different isolation levels and their effect on concurrency.

Related reference:

- Locks and concurrency control
- Deadlocks between applications [link to http://publib.boulder.ibm.com/infocenter/db2help/topic/com.ibm.db2.udb.doc/admin/c0005

Tracking lock waits and timeouts

A common user symptom of a locking problem is an application hang. A hang usually appears as ″Lock Wait″ within the database engine. To confirm that an application is in ″Lock Wait″, the database administrator should take a lock snapshot and an application snapshot.

When there is reason to believe that a locking conditions is being encountered the following steps are suggested:

Step 1. Take a lock snapshot

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

Step 2. Take an application snapshot

```
"db2 get snapshot for applications on sample"
```

Step 3. Get a list of applications connected to the database

```
"db2 list applications all"
```

The list of monitor elements that are of interest in such cases can be found here: Lock wait information monitor elements

Here is a truncated example of snapshot output:

```
Locks held by application = 46
Lock waits since connect = 12
Time application waited on locks (ms) = 96443
Deadlocks detected = 0
Lock escalations = 0
Exclusive lock escalations = 0
Number of Lock Timeouts since connected = 5
Total time UOW waited on locks (ms) = Not Collected
```

In the example, the application experienced 5 lock timeouts during its connection to the database. Also, we see that the application waited 96443 milliseconds for locks to be released. The average wait time can be calculated as follows:

```
(Time application waited for Locks / Lock waits since connect)
```

In the above case we see that each lock wait had an average time of 8036 milliseconds. In some environments (for example decision support systems) this might be considered normal, in others (for example online transaction processing) it would be considered excessive. Of course, user needs and business requirements should be used as measures.

In cases where you find lock timeouts becoming excessive when compared to normal operating levels, you may have an application that is holding locks for an extremely long period of time. It is also possible that if the number of lock timeouts is extremely low but the response time for queries is sluggish, the lock timeout parameter may be set too high.

Example 1: Identifying the cause of a lock timeout and lock wait

Please note, to simplify and not introduce unnecessary workload onto your environment, we’re going to model a long running query by disabling autocommit.

1. Open two DB2 command windows, and connect to sample from both windows
2. In the first window, issue the following query:
   a. `db2 +c ″insert into staff values (27,’Jones’,2,’Mgr’,13,35000,0)″`
3. From the second window issue the following query:
   a. `db2 ″select * from staff″`
4. Obtain a snapshot
   a. `db2 get snapshot for applications on sample`
5. View the snapshot
6. Issue a commit or a rollaback in both windows:
   a. `db2 commit`

Look for the ″`Application Status`″ field. You should see one of the applications in ″`UOW waiting`″ state, and the other in ″`Lock-wait`″ state. In the above case, you can see that application issuing the SELECT had made a lock request, but the application performing the insert was holding the requested lock. In the example, you had forced the application to never release locks until a commit, but the principle can be modeled to be the same in cases where locks are released after long periods of time.

This simple example demonstrates how to use a snapshot to find an application in Lock Wait. The next course of action may be to focus on what locks the application was waiting for and why those locks were not being released. If you re-do the example, but this time enable the monitor switches for locks (i.e. using ″`db2 update monitor switches using lock on`″) prior to performing the test, you will retrieve additional monitor elements such as ″`ID of agent holding lock`″, ″`Application ID holding lock`″, ″`Lock name`″, and ″`Lock mode requested`″.

Again, in this simplified case we actually forced the lock wait to happen, however when pursuing such issues, you may want to investigate the frequency of commits for the application, the locking semantics (isolation level) used or the size of an application's unit of work.

Related reference:

- Snapshot monitor sample output

Analyzing a lock snapshot

What follows is an example of how you might analyze snapshot output if you were investigating a lock wait scenario. To begin, open a lock snapshot that you have captured and search for an application status of ″`Lockwait`″. If you do not have one, you can create sample snapshots using the scenario described in **Tracking lock waits and timeouts**.

Extract Of A Snapshot For Application Handle 1 (with lock monitor switch enabled):

```
Application handle = 1
Application status = Lock-wait
...
Application country/region code = 1
DUOW correlation token = *LOCAL.DB2.050312015701
Application name = db2bp.exe
Application ID = *LOCAL.DB2.050312015701
...
Locks held by application = 4
Lock waits since connect = 2
Time application waited on locks (ms) = 12715
```

Take note of the application handle which is 1 in this example.

Determine the application that is holding the lock(s) that application handle 1 is waiting on:

```
.........................
ID of agent holding lock = 0
Application ID holding lock = *LOCAL.DB2.050312015656
........................
```

It can now be determined that the application with handle 1 is waiting on an application with handle 0. Determine the state of application handle 0.

```
...................
Application handle = 0
Application status = UOW Waiting
```

```
...
Application name = db2bp.exe
Application ID = *LOCAL.DB2.050312015656
...
```

Note that the application is in "UOW Waiting" state. This means that application is either doing some other processing or has an open UOW (unit of work). If the application is idle, i.e. there is no more work to do then the status will read "Idle". An idle application does not hold any database locks.

Examine the type of lock which application 1 is requesting. For example:

```
...............................
Lock object type = Row

Lock mode = Exclusive Lock (X)

Lock mode requested = Next Key Share (NS)

Name of tablespace holding lock = USERSPACE1

Schema of table holding lock = DB2

Name of table holding lock = STAFF

...................................
```

The lock snapshot shows that application handle 1 is requesting a Next Key Share (NS) lock but is waiting on application handle 0 that has an Exclusive Lock (X). Analyze the application snapshot to get more details about application handle 0. An (X) lock is not compatible with an (NS) lock, hence the Lock-Wait. The DB2 UDB Administration Guide outlines lock compatibilities.

Extract Of An Application Snapshot For Application Handle 0 (with uow and timestamp monitor switches enabled):

```
................
Elapsed time of last completed uow (sec.ms)= 0.000000
UOW start timestamp = 03/14/2005 12:16:07.872350
UOW stop timestamp =
UOW completion status =
................
```

If you have the statement monitor switch enabled at the time of the snapshot, you will also see the following:

```
................
Rows read = 2
Rows written = 1
Rows deleted = 0
Rows updated = 0
Rows inserted = 0
Rows fetched = 0
...
Blocking cursor = NO
Dynamic SQL statement text:
insert into staff values (27,'Jones',2,'Mgr',13,35000,0)
```

The application snapshot confirms that the UOW has begun but has not yet been completed. This indicates that the UOW is still open. The snapshot also tells us that the application handle is executing a dynamic SQL statement, an insert into the table STAFF in this case.

```
.............................
Number of SQL requests since last commit = 2
Commit statements = 0
Rollback statements = 0
Dynamic SQL statements attempted = 2
.............................
Locks held by application = 4
Lock waits since connect = 0
Time application waited on locks (ms) = 0
...
```

Application handle 0 has not yet committed and is holding 4 locks. Details about these locks can be seen in the lock snapshot, for example:

```
................................

List Of Locks

Lock Name = 0x02000300270000000000000052

Lock Attributes = 0x00000008

Release Flags = 0x40000000

Lock Count = 1

Hold Count = 0

Lock Object Name = 39

Object Type = Row

Tablespace Name = USERSPACE1

Table Schema = DB2

Table Name = STAFF

Mode = X
..................................
```

One of the 4 locks that application handle 0 is holding is an X lock. This lock name is the same as the one which the lock snapshot shows application 1 to be waiting for.

From the above analysis it can be determined that the cause of this locking condition is that an application is executing an INSERT into a table which requires exclusive access and has been granted all the required locks but has not committed. Any application with a weaker lock will have to wait for the UOW to commit. (This is the default behavior for the Cursor Stability (CS) isolation level.) Application handle 0 should commit more frequently.

Identifying the owner of a lock via db2pd

Click on this link for more information

Performance issues

Performance problems cover a wide range of scenarios:

- Identifiable query performing slower than expected
- Workload or batch job not completing as soon as expected, reduction in transaction rate or throughput

- Overall system slowdown
- Suspected bottleneck in some type of system resource such as CPU, I/O, memory
- Query or workload consuming more resource than expected or available
- Comparison is being made between one system and another
- Query, application, DB2, or system hangs

There are some subtleties in the scenarios depicted above. For problem diagnosis purposes, it is important to clarify whether something is not meeting expectations or is exceeding resource capacity. Sometimes it is both.

For problem determination purposes, hangs can be lumped together with performance problems because many investigative strategies apply to both. In addition, it may be not be possible at first to define the problem as a hang versus a performance problem. To a user waiting for a response, a long-running job can look like a hang even if in fact much activity can be taking place on behalf of the application on the database server. There can also be a significant buildup of activity during a severe system slowdown such that all or most commands appear to hang on a system.

In addition to characterizing the problem correctly in terms of where the symptom is observed (query/application/system resource) and what is wrong with it (slowness or too much resource used), you require many other pieces of information to put the problem in context. The following questions serve to quickly determine the best place to start looking for a potential cause.

When did the problem first occur?

If the problem has been occurring for some time, and if a database monitor schedule has been implemented, you can use historical data to find differences. This will allow you to focus on changes in system behavior and then focus on why these changes were introduced. Refer to Proactive monitoring tools. It is also important to consider whether any recent changes occurred, such as hardware or software upgrades, a new application rollout, additional users, etc.

Is the performance issue constant or intermittent?

If the poor performance is continual, check to see if the system has started to handle more load or if a shared database resource has become a bottleneck. Other potential causes of performance degradation include increased user activity, multiple large applications, or removal of hardware devices. If performance is poor only for brief periods begin by looking for common applications or utilities running at these times. If users report that a group of applications are experiencing performance issues, then you can begin your analysis by focusing on these applications.

Does the problem appear to be system-wide or isolated to DB2 and its applications?

System-wide performance problems suggest an issue outside of DB2. It's possible that something at the operating system level needs to be addressed.

If isolated to one application, is there a particular query that appears to be problematic?

If one application is problematic, then you can further examine if users are reporting a query or set of queries that are experiencing a slowdown. You might be able to isolate the issue to one application and a potential group of queries.

Is there any commonality to the poor performance or does it appear to be random?

You should determine if any common database tables, table space, indexes, etc. are involved. If so, this suggests that these objects are a point of contention. Other areas to potentially focus on are referential integrity constraints, foreign key cascades, locking issues etc.

Looking at DB2:

It's possible that after your initial review, you identify that there's a strong possibility that DB2 is the source of the performance issues. In this case, ask the following questions:

Are users reporting abnormally long response time for all applications?

1. Have you detected a large amount of I/O? Here are some possible causes:
   a. Overloaded devices.
   b. Heavy SORTS or use of temporary table spaces.
   c. Table scan of a large table.
   d. Insufficient buffer pool.
2. Have you detected a high CPU usage? Possible causes include:
   a. Heavy SORTS.
   b. Insufficient buffer pool sizes.
3. Both (1) and (2)
   a. Increased number of users?
   b. Large spilled SORTS.
   c. Application design?
4. Neither (1) and (2)?
   a. Are applications waiting on locks? Refer to Locking issues.
   b. Are applications waiting on data?

Are users reporting abnormally long response time for one application or query? Possible causes include:

- SORTS.
- Concurrency issues.
- Stale statistics.

These are general questions to ask yourself when faced with a performance issue. This list is by no means exhaustive given the size and complexity of modern systems, performance issues can surface from a multitude of possibilities. Regardless, these questions will start you on a logical path to determining where the problem came from.

Related reference:

- Elements of performance [Link to]
- Developing a performance improvement process [Link to]
- Performance tuning guidelines [link to]

Data To Collect For Optimizer Problems

Click on this link for more information

Analyzing access plans

Before analyzing an access plan, there are three basic things to note:

1. **Configuration parameters**: The optimizer considers many items when generating the most efficient access plan. Refer to Configuration parameters that affect query optimization to ensure that you are aware of the configuration choices that you have made which will affect the optimizer output. Effectively designing and tuning a system is key to improved access plans - and faster query run times.
2. **Optimization classes**: When an SQL query is compiled by DB2, a number of optimization techniques are used. Using more optimization techniques increases compile times and potentially system resource usage. Depending on the nature of the environment, it may be necessary to limit the number of techniques applied to optimizing the query. The methods by which you can set the optimization class can be found here: Setting the optimization class. If you know that the statements being executed within an application are relatively simple, for example that there are few joins, you may not want to use too many resources to optimize for higher speed. That is, you may find that the run-time differences between using a lower optimization level versus a higher one are minimal. If you have a case where extremely complex queries are being executed, a higher optimization level may be desirable, as more potentially cost effective access paths will be considered.
3. **Database statistics**: The DB2 optimizer bases its decisions based on database statistics that are stored in the system catalog. As such, it is good practice to ensure the statistics are current for the database. RUNSTATS is a DB2 utility that allows you to update table statistics. See RUNSTATS Command.

Viewing the access plan:

There are three methods of viewing an access plan. Refer to Explain tools. Note that prior to using these tools, you must create the EXPLAIN tables for the database. Refer to Explain tables.

To better understand access plans, you need to first understand these basic components:

```
  Cardinality
 <PLAN OPERATOR>
     Cost
   I/O Cost
```

The card or cardinality represents the number of rows that DB2 had estimated to be the output of the operator. The cost represents the CPU cost of this and previous operations, and IO cost represents the cost of the operator on the system's IO subsystems. For example, a snippet from db2exfmt output may appear as follows:

```
383.332 <- Cardinality
MSJOIN <- Plan Operator
( 5)
448.972 <- Cost
14.2765 <- I/O Cost
/---+---\
```

For an overview of the output from explain tools, refer to Description of db2expln and dynexpln output.

Tracking cardinality - how to measure its accuracy:

You may often find yourself reviewing an access plan that suggests that only a few rows will be returned. You need to examine and evaluate the cardinality estimates to determine how accurate these really are. For example:

```
                             100
    FETCH
                /-------------+-----------\
                100                       perfdb
                ISCAN                     Table1
                |                         1000
          Index: perfdb
                Index1
                1000
```

The above plan fragment indicates that the FETCH operation will return an expected 100 rows. If you suspect that more than 100 rows should be returned (or less. the principle is the same), you can determine and compare the ″true cardinality″ with the estimated. To do so, you need to first isolate the predicates applied at the ISCAN. You can find these in the detailed operator section of the access plan (this has been reduced to show only the important portions):

```
3) IXSCAN: (Index Scan)
Cumulative Total Cost: 30.9743
Cumulative CPU Cost: 149356
Cumulative I/O Cost: 1
Cumulative Re-Total Cost: 4.15582
Cumulative Re-CPU Cost: 103895
Cumulative Re-I/O Cost: 0
Cumulative First Row Cost: 30.9054
Estimated Bufferpool Buffers: 2

Predicates:
----------
2) Start Key Predicate
Relational Operator: Equal (=)
Subquery Input Required: No
Filter Factor: 0.1
Predicate Text:
--------------
(Q2.column1 = 5)
```

Taking the predicate applied here, you can formulate an SQL statement to test the validity of the cardinality estimate:

```
db2 select count(*) from table1 where column1=5
```

If you find that the output of the above query generates a value significantly different than the estimated value, the statistics may have become stale and a RUNSTATS may be necessary.

Tracking sorts that spilled to disk:

Another useful technique of isolating query bottlenecks is to determine if a sort spilled. This is an extension of the techniques developed for verifying cardinalities. Examine the change in operator costing. Here is an example:

```
3.65665e+07
TBSCAN
( 15)
```

```
6.87408e+06
1.45951e+06
|
3.65665e+07
SORT
( 16)
6.14826e+06
TBSCAN
( 17)
2.00653e+06
1.14206e+06
```

You see that from the SORT(16) operator to the TBSCAN(15) operator, the cost went from 6.14e6 to 6.87e6. This jump in COST for the operation directly following a sort indicates that the SORT has spilled to disk, and has created temporary tables in order to handle overflow rows. To determine the estimated number of pages expected to spill, jump to the operator details section of the plan:

```
15) TBSCAN:   (Table Scan)
.
.
.
Estimated Bufferpool Buffers: 163976
```

The estimated buffer pool buffers indicate that 163976 pages were expected to spill. Possible corrections include increasing sortheap or defining an index.

Related reference:

- Description of db2expln and dynexpln output
- Explain tools
- Examples of db2expln and dynexpln output
- Guidelines for analyzing explain information
- The SQL compiler process
- Guidelines for using explain information
- Optimization class guidelines

Tuning practices

Click on this link for more information

Sort overflows

Click on this link for more information

DB2 Connect performance troubleshooting

Click on this link for more information

Problem determination particular to multi-partition environments

In a multi-partition environment, the way to identify the partitions that are involved in a problem varies from problem to problem:

- Unexpected messages or SQL codes. In a partitioned database environment, the database partition where an application is submitted is the *coordinator partition*. Whenever a DB2 operation fails, the error message is returned to the coordinator partition. Then, based on the nature of the operation and diagnostic data collected from the coordinator partition, you can identify the rest of the nodes involved in this failed operation so that you can gather further diagnostic data from them.

- Abends (Abnormal ending). In a partitioned database environment, a DB2 instance could abend on all partitions or on some of the partitions. If you are not sure which partitions were shut down abnormally, check the DIAGPATH directories to see whether any trap files were generated at the specific time of the problem and where the trap files reveal the node numbers of partitions where the abend occurred. Or you may look through the db2diag.log files to search for the related *EDUCodeTrapHandler* entries, which are written into db2diag.log whenever there is an instance crash incident. The db2diag.log entry shows the node number information as well.
- Database or data corruption. Normally the first time you notice a database corruption problem is when you fail to connect to a certain database partition or fail to access a certain database object. You can immediately conclude that the partition (or partitions) where the database object resides probably has the corruption. If you are able to schedule a long enough window, you may further use the **db2dart** utility to inspect all or part of the partitions and accurately identify the corrupted partitions.
- Loops and hangs. When a specific operation or set of operations hangs, you might take a few application snapshots one minute apart on each partition, in order to determine what the status of the application is and whether any work is really being done. If the application status is "executing" but no counters are increasing, then that's the partition you need to focus on.
- Slow performance. For an operation with slow performance, you might again take a few application snapshots one minute apart on each partition, in order to determine what the status of the application is and how the work is being done. If the status is "executing" and counters like rows-read or written are increasing, you might be able to evaluate the performance based on how fast the counters increase. Some DB2 utilities provide query commands, such as `load query` and `rollforward` query, which you can use to identify the partitions where performance is slower than you expect.

Related reference:

- Data partitioning
- Recovering from transaction failures in a partitioned database environment

Common problems in multi-partition environments

Partitioned instance set-up

When you are using DB2 Extended Enterprise Server Edition, there are some additional steps that you must take if you wish to use multiple database partitions. Refer to Enabling data partitioning in a database for an overview of the considerations.

If you find that you cannot start the instance successfully after enabling data partitioning, some of the common causes are:

- The .rhosts file has not been configured properly, on UNIX and Linux platforms
- The fast communication manager (FCM) ports have not been reserved

Examples of these problem scenarios are provided below.

Problem 1: .rhosts file has not been configured

On UNIX and Linux platforms, when you have set up a partitioned instance and are trying to start the instance for the first time, you may encounter an SQL6048 error. For example:

```
%db2start
03-17-2005 18:13:32     0   0   SQL6048N  A communication error occurred during
START or STOP DATABASE MANAGER processing.
03-17-2005 18:13:34     1   0   SQL6048N  A communication error occurred during
START or STOP DATABASE MANAGER processing.
SQL1032N  No start database manager command was issued.  SQLSTATE=57019
```

If the .rhosts file has not been properly configured, you will see an indication of the source of the problem in the db2diag.log file:

```
2005-03-17-18.15.30.810022   Instance:db2inst1   Node:000
PID:59060(db2start)   TID:1   Appid:none
oper system services  sqloPdbExecuteRemoteCmd Probe:50

0xF01FD223 : ...            myhost.ibm.com

2005-03-17-18.15.30.829173   Instance:db2inst1   Node:000
PID:59060(db2start)   TID:1   Appid:none
oper system services  sqloPdbExecuteRemoteCmd Probe:50

0x2FF1D75C : 7273 6864 3A20 3038 3236 2D38 3133 2050    rshd: 0826-813 P
0x2FF1D76C : 6572 6D69 7373 696F 6E20 6973 2064 656E    ermission is den
0x2FF1D77C : 6965 642E 0A                               ied..
...
```

Since DB2 ESE uses the rsh command to execute some commands such as db2start on all of the partitions, each partition must have the authority to perform remote commands on the other partitions. This can be done by updating the .rhosts file in the instance home directory. Refer to Enabling the execution of remote commands (UNIX).

After you edit the .rhosts file properly, run db2start again. It should now be successful on all partitions.

Problem 2: Too few (or no) FCM ports have been reserved

This situation can occur on all platforms. It is usually typified by a SQL6031 error on db2start. For example:

```
SQL6031N  Error in the db2nodes.cfg file at line number "3".  Reason code "12".
```

To find out what reason code 12 refers to, execute the db2 ? command:

```
% db2 ? SQL6031
...
(12) The port value at line "<line>" of the db2nodes.cfg file in
the sqllib directory is not in the valid port range defined for
your DB2 instance id in the services file (/etc/services on
UNIX-based systems).
...
```

In a partitioned database environment, most communication between database partitions is handled by the Fast Communications Manager (FCM). To enable the FCM at a database partition and allow communication with other database partitions, you must create a service entry in the /etc/services file. The FCM uses the specified port to communicate. If you have defined multiple partitions on the same host, you must define a range of ports. Refer to the appropriate link for further information:

- Verifying port range availability on participating computers (UNIX)
- Verifying port range availability on participating computers (Windows)

Related reference:

- Creating a node configuration file
- Determining problems with rah on UNIX-based platforms

Log full situation during redistribution: After adding a database partition and updating the nodegroup definition, you need to use the REDISTRIBUTE NODEGROUP utility to move the data to the appropriate database partitions. Partition redistribution is an insert-delete procedure. DB2 logs each insert and delete SQL on the related partitions. If the log space size is not big enough, it's possible for the partition redistribution operation to fail with an SQL0964 ("The transaction log for the database is full") error when a large number of records need to be redistributed across partitions. For example:

```
% db2 "redistribute nodegroup ibmdefaultgroup uniform"
SQL6064N SQL error "-964" occurred during data redistribution.
% db2 ? sql0964
SQL0964C The transaction log for the database is full.
```

You can confirm the error message in the db2diag.log file:

```
2005-03-18-13.02.47.734287-300 E13748A504         LEVEL: Error
PID     : 7176390              TID  : 1           PROC : db2agntp (SAMPLE) 0
INSTANCE: db2inst1              NODE : 000         DB   : SAMPLE
APPHDL  : 0-52                 APPID: *N0.lcawley.050318180122
FUNCTION: DB2 UDB, data protection, sqlpgResSpace, probe:2860
MESSAGE : ADM1823E  The active log is full and is held by application handle "52".  Terminate this
APPLICATION.

2005-03-18-13.02.47.737550-300 I14253A469         LEVEL: Error
PID     : 7176390              TID  : 1           PROC : db2agntp (SAMPLE) 0
INSTANCE: db2inst1              NODE : 000         DB   : SAMPLE
APPHDL  : 0-52                 APPID: *N0.lcawley.050318180122
FUNCTION: DB2 UDB, data protection, sqlpWriteLR, probe:6680
RETCODE : ZRC=0x85100009=-2062548983=SQLP_NOSPACE
          "Log File has reached its saturation point"
          DIA8309C Log file was full.
...
2005-03-18-13.02.47.827614-300 I15116A685         LEVEL: Error
PID     : 5865510              TID  : 1           PROC : db2agent (SAMPLE) 0
INSTANCE: lcawley              NODE : 000         DB   : SAMPLE
APPHDL  : 0-52                 APPID: *N0.lcawley.050318180122
DATA #1 : SQLCA, PD_DB2_TYPE_SQLCA, 136 bytes
 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: -964   sqlerrml: 0
 sqlerrmc:
 sqlerrp : sqlridel
 sqlerrd : (1) 0x85100009      (2) 0x00000009      (3) 0x00000000
           (4) 0x00000000      (5) 0x00000000      (6) 0x00000000
 sqlwarn : (1)      (2)      (3)      (4)        (5)       (6)
           (7)      (8)      (9)      (10)        (11)
 sqlstate:
```

From the db2diag.log file, you can tell that partition (NODE) 0 hit the log full error. You now need to increase the log space size on that partition. You can either increase the log file size or the number of the log files. The following example chooses to increase the log file size:

Switch to partition 0:

```
% db2 terminate
% export DB2NODE=0
```

Increase the log space size by increasing the LOGFILSIZ db cfg parameter:

```
% db2 update db cfg for sample using LOGFILSIZ 1000
```

Then you can retry the redistribute operation. If the data redistribution operation fails, some tables may be redistributed while others are not. This occurs because data redistribution is performed one table at a time. So you cannot recover from the failing redistribution operation by simply reissuing that command. Refer to Redistribution error recovery for more details.

Related reference:

- Log space requirements for data redistribution
- Redistributing data across partitions
- Redistribution error recovery

Dropping a database partition: Click on this link for more information.

SQL1035 (database in use) when taking offline backups in parallel: The version recovery method requires loading a backup copy of the database. The database will be restored to exactly the same state that it was in when it was backed up. Using the version recovery method, you must schedule and perform full backups of the database on a regular basis.

You may have the following experience: All database partitions are inactive at the moment. However, offline backup of all partitions in parallel fails with SQL1035N (The database is currently in use.) on most of the partitions.

You try to take offline backup of all partitions in parallel as follows:

```
% db2 force applications all
% db2_all ";db2 backup database sample to /database/backup"
host1: SQL1035N The database is currently in use. SQLSTATE=57019
host1: db2 backup database ... completed rc=4
host2: SQL1035N The database is currently in use. SQLSTATE=57019
host2: db2 backup database ... completed rc=4
host3: Backup successful. The timestamp for this backup image is :
20050318135804
host3: db2 backup database ... completed ok
```

You can confirm the error in the db2diag.log file:

```
2005-03-18-13.58.04.542575-300 I45063A686         LEVEL: Error
PID     : 8085644              TID  : 1           PROC : db2agent (SAMPLE) 1
INSTANCE: db2inst1              NODE : 001         DB   : SAMPLE
APPHDL  : 1-58                 APPID: *N1.db2inst1.050318185804
DATA #1 : SQLCA, PD_DB2_TYPE_SQLCA, 136 bytes
 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: -1035   sqlerrml: 0
 sqlerrmc:
 sqlerrp : SQLESUBC
 sqlerrd : (1) 0x00000000      (2) 0x00000000      (3) 0x00000000
           (4) 0x00000000      (5) 0x00000000      (6) 0x00000000
 sqlwarn : (1)      (2)      (3)      (4)        (5)       (6)
           (7)      (8)      (9)      (10)        (11)
 sqlstate:
```

This occurs because the database backup utility requires exclusive access to the catalog partition.

To properly backup the database in parallel, you may use the following commands:

```
% db2_all "<<+0<; db2 backup database sample to /database/backup"
% db2_all "<<-0<; db2 backup database sample to /database/backup"
```

This assumes that CATALOG PARTITION is partition 0. It is also a good illustration of why the CATALOG PARTITION should contain catalog data ONLY.

Related reference:

- BACKUP DATABASE command
- Version recovery

Chapter 3. Searching knowledge bases

DB2 troubleshooting resources

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2® products.

DB2 documentation:

Troubleshooting information can be found throughout the DB2 Information Center, as well as throughout the PDF books that make up the DB2 library. You can refer to the ″Support and troubleshooting″ branch of the DB2 Information Center navigation tree (in the left pane of your browser window) to see a complete listing of the DB2 troubleshooting documentation.

DB2 Technical Support Web site:

Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs), FixPaks and the latest listing of internal DB2 error codes, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at http://www.ibm.com/software/data/db2/udb/support

Chapter 4. Getting fixes

Release Notes

Information Integrator Release Notes

DB2 Release Notes

How to effectively search for known problems

There are many resources available that describe known problems, including DB2 APARs, whitepapers, redbooks, technotes and manuals. It is important to be able to effectively search these (and other) resources in order to quickly determine whether a solution already exists for the problem you are experiencing.

Before searching, it is important to have a clear understanding of your problem situation. If you do not, refer to the Introduction to problem determination.

Once you have a clear understanding of what the problem situation is, you need to compile a list of search keywords to increase your chances of finding the existing solutions. Here are some tips:

1. Use multiple words in your search. The more pertinent search terms you use, the better your search results will be.
2. Start with specific results, and then go to broader results if necessary. i.e. If too few results are returned, then remove some of the less pertinent search terms and try it again. Alternatively, if you are uncertain which keywords to use, you can perform a broad search with a few keywords, look at the type of results that you receive, and be able to make a more informed choice of additional keywords.
3. Sometimes it is more effective to search for a specific phrase. For example, if you enter: ″administration notification file″ (with the quotes) you will get only those documents that contain the exact phrase in the exact order in which you type it. (As opposed to all documents that contain any combination of those three words).
4. Use wildcards. If you are encountering a specific SQL error, search for ″`SQL5005<wildcard>`″, where `wildcard` is the appropriate wildcard for the resource you're searching. This is likely to return more results than if you had merely searched for ″`SQL5005`″ or ″`SQL5005c`″.
5. If you are encountering a situation where your instance ends abnormally and produces trap files, search for known problems using the first two or three functions in the trap or core file's stack traceback. If too many results are returned, try adding keywords ″`trap`″, ″`abend`″ or ″`crash`″.
6. If you are searching for keywords that are operating-system-specific (such as signal numbers or errno values), try searching on the constant name, not the value. i.e. Search for ″EFBIG″ instead of the error number 27. For information about matching operating system error numbers to their constant names, see Platform specific error logs and Formatting and interpreting trap files.

In general, search terms that are successful often involve:

- Words that describe the command run
- Words that describe the symptoms

- Tokens from the diagnostics

Related reference:

- Introduction to problem determination
- Platform specific error logs
- Formatting and interpreting trap files

Authorized Program Analysis Reports (APARs) and fix packs

A DB2 FixPak contains updates and fixes for problems (Authorized Program Analysis Reports, or ″APARs″). These APARs address problems found during testing at IBM, as well as fixes for problems reported by customers.

APARs have unique identifiers. For example, ″IY53671″ and ″JR19727″. Each DB2 APAR is specific to a particular version of DB2 (e.g. DB2 version 8 or DB2 version 7).

Every FixPak is accompanied by a document, called APARLIST.TXT, that lists the APARs it contains.

FixPaks are cumulative. This means that the latest FixPak for any given version of DB2 contains all of the updates from previous FixPaks for the same version of DB2. It is recommended that you keep your DB2 environment running at the latest FixPak level to ensure problem-free operation.

Viewing APAR content:

To get a better understanding of APARs, go to the DB2 UDB Support website for Linux, Windows and UNIX platforms:

http://www.ibm.com/software/data/db2/udb/support/

In the search bar, enter one of the APAR identifiers mentioned above, i.e. ″IY53671″ or ″JR19727″.

Here is what you should see for IY61009, for example:

```
IY61009: Cannot directly change a tablespace privilege from yes to grant.

 A fix is available

DB2 Universal Database Version 8 FixPak 8


APAR status
Closed as program error.


Error description
Users who have privileges to access a tablespace without the
grant option cannot be granted permission with the grant option
without having their privilege on the tablespace dropped first.
Users who are granted access with the grant option up front do
not have a problem.

Local fix

Drop the user's access to the tablespace and then reissue the
grant using the 'with grant option'.
```

```
Problem summary

Users Affected:      All

Problem Description:
Users who may think they have access to a tablespace with the
grant option will not be able to grant other users access to the
tablespace.

Catalog table SYSCAT.TBSPACEAUTH.USEAUTH will be 'Y' instead of
'G' for any user who was granted access to the tablespace
without the grant option and then granted access to the
tablespace with the grant option.

Problem Summary:
Users are unable to grant access to a tablespace when they have
been given explicit grant access with the grant option.

Problem conclusion

Problem was first fixed in Version 8.2 FixPak 8 (s041221)

Temporary fix


Comments


APAR information
APAR number IY61009
Reported component name DB2 UDB ESE AIX
Reported component ID 5765F4100
Reported release 810
Status CLOSED PER
PE NoPE
HIPER NoHIPER
Submitted date 2004-08-24
Closed date 2005-02-04
Last modified date 2005-02-09

APAR is sysrouted FROM one or more of the following:

APAR is sysrouted TO one or more of the following:

Modules/Macros
DEFSRXXX

Fix information
Applicable component levels
R810 PSY    UP
```

Look at various fields in the APAR.:

- The *abstract* is the title. It is a brief description of the defect.
- The *error description* describes in detail the symptoms experienced. Depending on the complexity of the problem, this information will allow you to confirm if you are encountering the defect described.
- Both the *local fix* and *temporary fix* describe circumvention (if available), and corrective steps to work around the defect (if available).
- The *problem summary* may go into some detail about what causes the problem.
- The *problem conclusion* typically indicates which FixPak will contain the APAR. This information is also specified under the heading ″Fixes are available″.

Open APARs:

Go back to the main page on the DB2 Support website. Click on the ″APARs″ link. It should take you to the following page:

http://www.ibm.com/software/data/db2/udb/support/apars.html

Here you will see lists of (among other things) ″Open APARs″. Open APARs are those which are currently being worked upon or are waiting to be included in the next available FixPak. You can view lists of all of these outstanding APARs on this website.

Open APARs are likely to be resolved in the next available FixPak for the version that they are opened against. There are two exceptions to this:

- an APAR might be identified too late in a FixPak's development and testing cycle to be included, or
- an APAR might be closed, to be fixed in a future release (closed as ″FIN″). The explanation of a FIN APAR is:

If a defect is determined to be of lower impact which does not require an immediate, permanent fix, [IBM] may defer the fix for a future release. APARs will reflect deferred fixes with a closing code of ″FIN ″(Fixed If there is a Next release) to designate plans for inclusion in a future release. -- *IBM Software Support Handbook* http://techsupport.services.ibm.com/guides/handbook.html

Normal vs. HIPER APARs:

On this web page (http://www.ibm.com/software/data/db2/udb/support/apars.html), you can also see information about ″HIPER APARs″.

High-Impact PERvasive (HIPER) APARs are critical bugs of which all DB2 customers should be aware. Bugs that might result in data loss, system outages, function loss, poor performance, or are pervasive in the field are typically categorized as HIPER APARs.

A HIPER APAR has the same format as all other APARs. If you click on one of these HIPER APARs, though, you will see that there is a flag at the bottom of the APAR information:

```
Yes HIPER Yes HIPER
```

What to do about HIPER APARs:

You should review each HIPER APAR that is not resolved in the DB2 version and FixPak that you are running. This helps you assess the risk exposure of staying at a particular FixPak level.

Reading the abstract of the less severe open APARs also helps you assess the risk.

Related reference:

- Applying the latest FixPak (Windows and UNIX)

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 Information Integrator
DB2 Query Patroller
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.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Index

Contacting IBM

In the United States, call one of the following numbers to contact IBM:

- 1-800-IBM-SERV (1-800-426-7378) 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 http://www.ibm.com/planetwide

Product information

Information regarding DB2 Universal Database products is available by telephone or by the World Wide Web at http://www.ibm.com/software/data/db2/udb

This site contains the latest information on the technical library, ordering books, product 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 USA

Spine information:

IBM® DB2® Universal Database DB2 Troubleshooting Guide