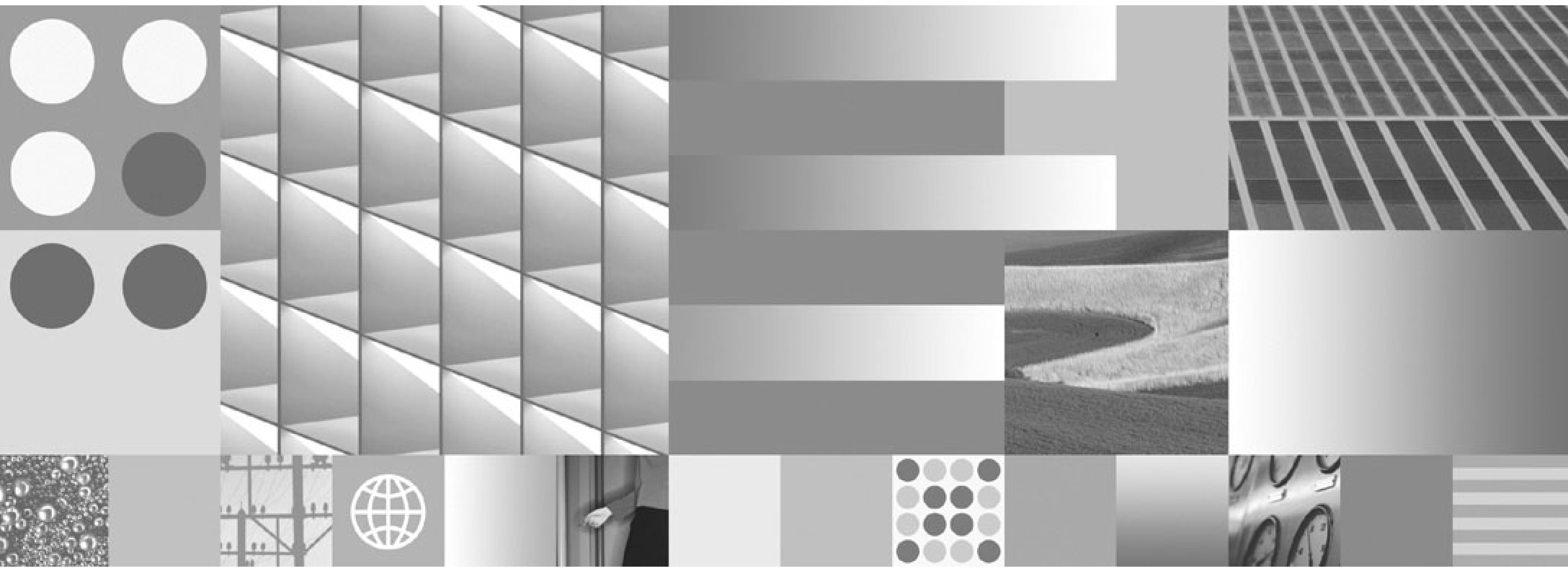
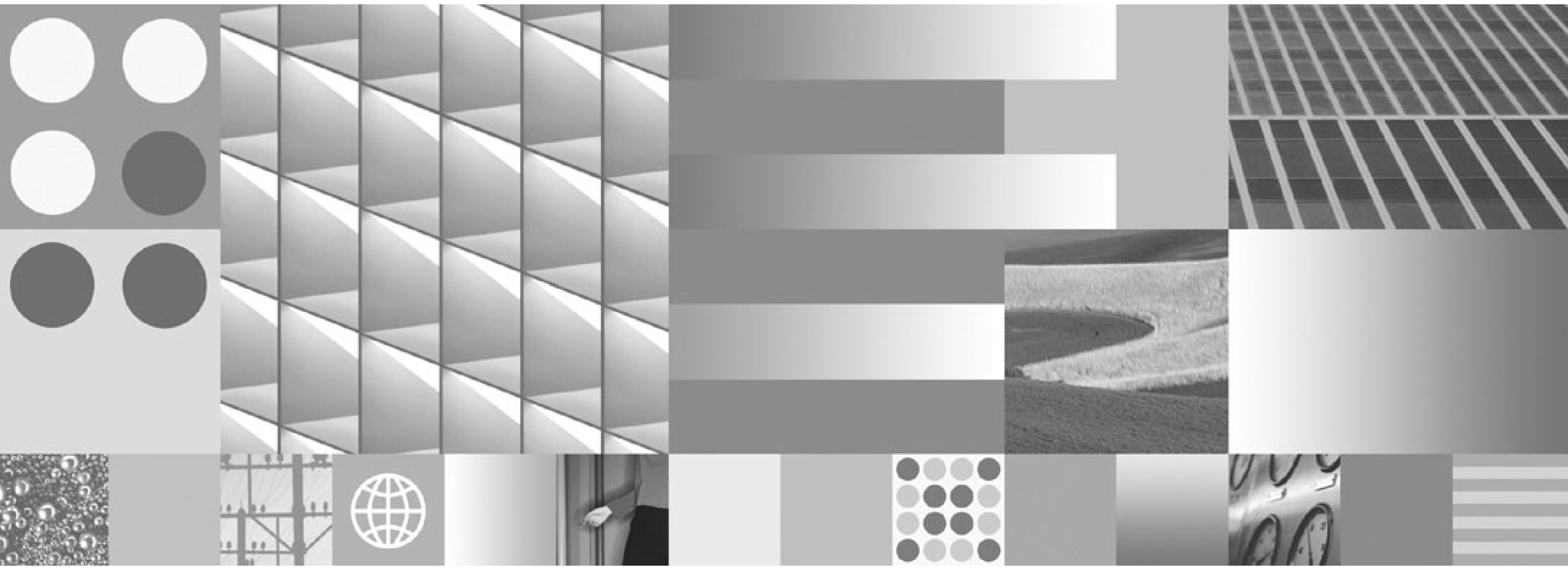
DB2 Version 9.5
for Linux, UNIX, and Windows



Troubleshooting Guide
Updated March, 2008

GI11-7857-01

DB2 Version 9.5
for Linux, UNIX, and Windows



Troubleshooting Guide
Updated March, 2008

GI11-7857-01

Note

Before using this information and the product it supports, read the general information under Appendix B, “Notices,” on page 113.

Edition Notice

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

About this book

This guide provides information to get you started solving problems with DB2® database clients and servers. It helps you to:

- Identify problems and errors in a concise manner
- Solve problems based on their symptoms
- Learn about available diagnostic tools
- Develop a troubleshooting strategy for day-to-day operations.

Who should use this book?

This guide is intended for customers, users, system administrators, database administrators (DBAs), communication specialists, application developers, and technical support representatives for DB2 database clients and servers. To use it, you should be familiar with:

- Communications, relational database, and local area network (LAN) concepts
- Hardware and software requirements and options
- The overall configuration of your network
- Application programs and other facilities that run on your network
- Basic DB2 database administrative tasks
- The information on installation and early tasks described in the *Quick Beginnings* guides for the products you have installed.

Chapter 1. Learning more about troubleshooting

At some point when working with DB2 database products, you may encounter a problem. This problem might be reported by the database manager, by an application running against the database, or by your users as they give feedback to you that “something is not quite right” with the database. The concepts and tools presented here are to introduce you to, and to help you with, the task of troubleshooting a real or perceived problem in the operations of your database. The importance of capturing the right data at the right time is emphasized and so first occurrence data capture is the first tool discussed. Other logs and files that are used by the database manager to capture data about the operations of the database are presented including mention of operating system diagnostic tools.

Introduction to troubleshooting

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

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

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

What are the symptoms?

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

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

Where is the problem happening?

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

- Is the problem platform specific, or common to multiple platforms?
- Is the current environment and configuration supported?
- Is the application running locally on the database server or on a remote server?
- Is there a gateway involved?

- Does the database reside on individual disks, or on a RAID disk array?

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

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

When does the problem happen?

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

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

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

Under which conditions does the problem happen?

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

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

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

Is the problem reproducible?

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

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

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

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

About first occurrence data capture

First occurrence data capture (FODC) is the term applied to the set of diagnostic information that DB2 database products capture automatically when errors occur. The information reduces the need to reproduce errors to get diagnostic information.

The topics concerned with FODC introduce the importance of diagnosis information and the different methods to gather it. There are several logs that are associated with database tools, utilities, and DB2 products that will be useful to you when troubleshooting a problem with your database.

Collecting diagnosis information based on common outage problems

Diagnostic information can be gathered automatically in a package when an outage occurs affecting an instance. The information in the package can also be created manually.

When trouble occurs when working with DB2 instances and databases, you should collect data at the time the problem happens. First occurrence data collection (FODC) is the term used to describe the actions taken when trouble occurs in your DB2 environment. You control what data is collected during outages through the setting of options in the DB2FODC registry variable using the db2pdcfg tool. Use db2pdcfg -fodc to change the DB2FODC registry variable options. The options influence the database system behavior regarding data capture in FODC situations.

Automatic collection of diagnostic information

The database manager invokes the db2fodc command for automatic First Occurrence Data Capture (FODC).

To correlate the outage with the DB2 diagnostic logs and the other troubleshooting files, a diagnostic message is written to both the administration notification log and the db2diag.log. The diagnostic message includes the FODC directory name and the timestamp when the FODC directory was created. The FODC package description file is placed in the new FODC directory.

Manual collection of diagnostic information

The first occurrence data collection command (db2fodc) is used to collect information about potential hangs, or when there are severe performance issues. When the db2fodc command is run, a new directory FODC_hang_<timestamp> is automatically created under the current diagnostic path. The db2cos_hang script is run. This script controls the data collection that will be gathered and placed in the FODC subdirectories. The existence of the FODC subdirectories depends on the way the db2fodc command is run or on the configuration of the DB2 registry variable.

Configuring for automatic collection of diagnostic information

Before the database manager can carry out actions automatically, you have to indicate to the database manager what actions are to be taken.

Flags are set indicating actions to be taken by the database manager when an error or a warning is encountered during database operations. The actions that are carried out include:

- Producing a stack trace in the db2diag.log. (Default)
- Running the callout script, db2cos. (Default)
- Stopping the trace (db2trc) command.

Change the first occurrence data capture (FODC) options

Change the first occurrence data capture (FODC) options using the Configure DB2 database for problem determination behavior (db2pdcfg) command. The FODC options are set in the DB2FODC registry variable using the db2pdcfg tool. The options influence the database system behavior regarding data capture in FODC situations.

Data collected as part of FODC and its placement

Depending on the type of outage within the instance, first occurrence data capture (FODC) results in the creation of subdirectories and specific content that is collected. A series of subdirectories is created along with the collection of files and logs.

One or more of the following subdirectories is created under the FODC directory:

- DB2CONFIG containing DB2 configuration output and files
- DB2PD containing db2pd output or output files
- DB2SNAPS containing DB2 snapshots
- DB2TRACE containing DB2 traces
- OSCONFIG containing operating system configuration files
- OSSNAPS containing operating system monitor information
- OSTRACE containing operating system traces

These directories may not always exist depending on the configuration of DB2FODC or the mode in which db2fodc is run.

Depending on the type of outage, the following content is found in the FODC directory and subdirectories:

- Trap files

- All the different binary and plain text dump files generated during the data capture as part of the outage and completed by different components
- db2evlog's event log file
- DB2 trace dump if trace has been on at the time of the outage
- The directory containing the core file
- DB2FODC log files:
  - Only one "log" file is used for a manual FODC. db2fodc_hang.log (for hangs) or db2fodc_badpage.log (for bad pages)
- Data corruption related information
  - Process information: ps (on UNIX) and db2pd -edus output
  - Additional information collected currently by db2support (optional):
    - errpt -a output (on AIX)
    - System logs on UNIX platforms. For example, /var/adm/messages for SunOS and /var/adm/syslog.log on HP/UX. This will be done provided that these files may be collected (on Linux, you must have root access to copy the syslog file).

Automatic FODC data generation

When an outage occurs and automatic first occurrence data capture (FODC) is enabled, data is collected based on symptoms. The data collected is specific to what is needed to diagnose the outage.

One or many messages, including those defined as "critical" are used to mark the origin of an outage.

Trap files contain information such as:
- The amount of free virtual memory
- Values associated with the product's configuration parameters and registry variables at the time the trap occurred
- Estimated amount of memory used by the DB2 product at the time of the trap
- Information that provides a context for the outage

The raw stack dump may be included in an ASCII trap file.

Dump files that are specific to components within the database manager are stored in the appropriate FODC package directory.

First occurrence data capture information

First occurrence data capture (FODC) is the process used to capture scenario-based data about a DB2 instance. FODC can be invoked manually by a DB2 user based on a particular symptom or invoked automatically when a predetermined scenario or symptom is detected.This information reduces the need to reproduce errors to get diagnostic information.

FODC information can be found in the following files:

Administration notification log ("*instance_name*.nfy")
- Operating system: All
- Default location:
  - Linux® and UNIX®: Located in the directory specified by the *diagpath* database manager configuration parameter.

  - Windows®: Use the Event Viewer Tool (Start>Control Panel>Administrative Tools >Event Viewer)
- Created automatically when the instance is created.
- When significant events occur, DB2 writes information to the administration notification log. The information is intended for use by database and system administrators. The type of message recorded in this file is determined by the *notifylevel* configuration parameter.

DB2 diagnostic log (″db2diag.log″)

- Operating system: All
- Default location: Located in the directory identified by the *diagpath* database manager configuration parameter.
- Created automatically when the instance is created.
- This text file contains diagnostic information about error and warnings encountered by the instance. This information is used for troubleshooting and is intended for IBM® software support. The type of message recorded in this file is determined by the *diaglevel* database manager configuration parameter.

DB2 administration server (DAS) diagnostic log (″db2dasdiag.log″)

- Operating system: All
- Default location:
  - Linux and UNIX: Located in DASHOME/das/dump, where DASHOME is the home directory of the DAS owner
  - Windows: Located in ″dump″ folder, in the DAS home directory. For example: C:\Program Files\IBM\SQLLIB\DB2DAS00\dump
- Created automatically when the DAS is created.
- This text file contains diagnostic information about errors and warnings encountered by the DAS.

DB2 event log (″db2eventlog.xxx″, where xxx is the database partition number)

- Operating system: All
- Default location: Located in the directory specified by the *diagpath* database manager configuration parameter
- Created automatically when the instance is created.
- The DB2 event log file is a circular log of infrastructure-level events occurring in the database manager. The file is fixed in size, and acts as circular buffer for the specific events that are logged as the instance runs. Every time you stop the instance, the previous event log will be replaced, not appended. If the instance traps, a db2eventlog.XXX.crash file is also generated. These files are intended for use by IBM software support.

DB2 callout script (db2cos) output files

- Operating system: All
- Default location: Located in the directory specified by the *diagpath* database manager configuration parameter
- Created automatically when a panic, trap or segmentation violation occurs. Can also be created during specific problem scenarios, as specified using the db2pdcfg command.

- The default db2cos script will invoke db2pd commands to collect information in an unlatched manner. The contents of the db2cos output files will vary depending on the commands contained in the db2cos script.
- The db2cos script is shipped under the bin/ directory. On UNIX, this directory is read-only. To create your own modifiable version of this script, copy the db2cos script to the adm/ directory. You are free to modify this version of the script. If the script is found in the adm/ directory, it is that version that is run. Otherwise, the default version in the bin/ directory is run.

Dump files

- Operating system: All
- Default location: Located in the directory specified by the *diagpath* database manager configuration parameter
- Created automatically when particular problem scenarios arise.
- For some error conditions, extra information is logged in binary files named after the failing process ID. These files are intended for use by IBM software support.

Trap files

- Operating system: All
- Default location: Located in the directory specified by the *diagpath* database manager configuration parameter
- Created automatically when the instance ends abnormally. Can also be created at will using the db2pd command.
- The database manager generates a trap file if it cannot continue processing due to a trap, segmentation violation, or exception.

Core files

- Operating system: Linux and UNIX
- Default location: Located in the directory specified by the *diagpath* database manager configuration parameter
- Created by the operating system when the DB2 instance terminates abnormally.
- Among other things, the core image will include most or all of DB2's memory allocations, which may be required for problem descriptions

To improve troubleshooting as part of trap, panic, data corruption, or hang outage situations (when the database or instance cannot be used), DB2 diagnostic files related to the specific outage type will be directed to a new First Occurrence Data Capture (FODC) directory. This directory is created under the instance diagnostic path.

Note: This FODC directory is named FODC_<symptom>_<timestamp>. The DB2 diagnostic files will be in a FODC package. Only AIX and Linux operating systems have this directory and package creation.

DB2 Query Patroller and First Occurrence Data Capture (FODC)

If you find you have a need to investigate DB2 Query Patroller problems, there are logs that contain information about the probable cause for the difficulties you may be experiencing.

qpdiag.log

- Operating system: All
- Default location: Located in the directory identified by the *diagpath* database manager configuration parameter.
- Created automatically when the Query Patroller system becomes active.
- Contains informational and diagnostic records for Query Patroller. This information is used for troubleshooting and is intended for use by IBM Software Support.

qpmigrate.log

- Operating system: All
- Default location: Located in the directory identified by the *diagpath* database manager configuration parameter.
- Created automatically by the qpmigrate utility. The qpmigrate command can be run implicitly when you install Query Patroller (if you specify an existing database to run Query Patroller on), or explicitly after the installation.
- Captures information and error messages when Query Patroller is migrated from one version to another. It is intended for use by Query Patroller administrators.

qpsetup.log

- Operating system: All
- Default location: Located in the directory identified by the *diagpath* database manager configuration parameter.
- Created automatically by the qpsetup utility. The qpsetup command can be run implicitly when you install Query Patroller (if you specify an existing database to run Query Patroller on), or explicitly after the installation.
- Captures information and error messages that occur while the qpsetup utility is running. It is intended for use by Query Patroller administrators.

qpuser.log

- Operating system: All
- Default location: Located in the directory identified by the *diagpath* database manager configuration parameter.
- Created automatically when the Query Patroller system becomes active.
- Contains informational messages about Query Patroller; for example, indicating when Query Patroller stops and starts. It is intended for use by Query Patroller administrators.

Monitor and audit facilities using First Occurrence Data Capture (FODC)

If you find you have a need to investigate monitor or audit facility problems, there are logs that contain information about the probable cause for the difficulties you may be experiencing.

DB2 audit log (″db2audit.log″)

- Operating system: All
- Default location:
  - Windows: Located in the $DB2PATH*instance_name*\security directory

  – Linux and UNIX: Located in the $HOME\sqllib\security directory, where $HOME is the instance owner's home directory
- Created when the db2audit facility is started.
- Contains audit records generated by the DB2 audit facility for a series of predefined database events.

DB2 governor log ("*mylog.x*", where *x* is the number of database partitions on which the governor is running)

- Operating system: All
- Default location:
  – Windows: Located in the $DB2PATH\instance_name\log directory
  – Linux and UNIX: Located in the $HOME\sqllib\log directory, where $HOME is the instance owner's home directory
- Created when using the governor utility. The base of the log file name is specified in the db2gov command.
- Records information about actions performed by the governor daemon (for example, forcing an application, reading the governor configuration file, starting or ending the utility) as well as errors and warnings.

Event monitor file (for example, "00000000.evt")

- Operating system: All
- Default location: When you create a file event monitor, all of the event records are written to the directory specified in the CREATE EVENT MONITOR statement.
- Generated by the event monitor when events occur.
- Contains event records that are associated with the event monitor.

Graphical tools using First Occurrence Data Capture (FODC)

If you find you have a need to investigate command editor, Data Warehouse Center, or Information Catalog Center problems, there are logs that contain information about the probable cause for the difficulties you may be experiencing.

Command Editor log

- Operating system: All
- Default location: The name and location of this log file are specified using the Command Editor page of the DB2 toolbar. If a path is not specified, the log is stored in the $DB2PATH\sqllib\tools directory on Windows and in the $HOME/sqllib/tools directory on Linux and UNIX, where HOME is the instance owner's home directory.
- Created when you select **Log command history to file** in the Command Editor and then specify the file and location.
- Contains the command and statement execution history from the Command Editor.

Data Warehouse Center IWH2LOGC.log file

- Operating system: All
- Default location: Located in the directory that is specified by the VWS_LOGGING environment variable. The default path is the $DB2PATH\sqllib\logging directory on Windows and in the $HOME/sqllib/logging directory on Linux and UNIX, where HOME is the instance owner's home directory
- Created automatically by the Data Warehouse Center if the logger stops.

- Contains messages written by the Data Warehouse Center and OLE server that could not be sent in the situation where the logger stops. This log may be viewed using the Log Viewer window in the Data Warehouse Center.

Data Warehouse Center IWH2LOG.log file

- Operating system: All
- Default location: Located in the directory that is specified by the VWS_LOGGING environment variable. The default path is the $DB2PATH\sqllib\logging directory on Windows and in the $HOME/sqllib/logging directory on Linux and UNIX, where HOME is the instance owner's home directory
- Created automatically by the Data Warehouse Center when it cannot start itself or when a trace is activated.
- Contains diagnostic information for situations when the Data Warehouse Center logger cannot start itself and cannot write to the Data Warehouse Center log (IWH2LOGC.log). This log may be viewed using the Log Viewer window in the Data Warehouse Center.

Data Warehouse Center IWH2SERV.log file

- Operating system: All
- Default location: Located in the directory that is specified by the VWS_LOGGING environment variable. The default path is the $DB2PATH\sqllib\logging directory on Windows and in the $HOME/sqllib/logging directory on Linux and UNIX, where HOME is the instance owner's home directory
- Created automatically by the Data Warehouse Center server trace facility.
- Contains Data Warehouse Center start up messages and logs messages created by the server trace facility. This log may be viewed using the Log Viewer window in the Data Warehouse Center.

Information Catalog Center tag file EXPORT log

- Operating system: All
- Default location: The exported tag file path and log file name are specified in the **Options** tab of the Export tool in the Information Catalog Center
- Generated by the Export tool in the Information Catalog Center
- Contains tag file export information, such as the times and dates when the export process started and stopped. It also includes any error messages that are encountered during the export operation.

Information Catalog Center tag file IMPORT log

- Operating system: All
- Default location: The imported tag file path and log file name are specified in the Import tool in the Information Catalog Center
- Generated by the Import tool in the Information Catalog Center
- Contains tag file import history information, such as the times and dates when the import process started and stopped. It also includes any error messages that are encountered during the import operation.

About administration notification log files

An introduction to administration notification log files describing the different levels of the type of errors that are logged and how to establish an error capture level. Once the logs are created, guidance is given on how to interpret the log records so that you may understand more about your specific database problem.

Administration notification log

The DB2 database manager writes the following kinds of information to the administration notification log: the status of DB2 utilities such REORG and BACKUP; client application errors; service class changes, licensing activity; log file paths and storage problems; monitoring and indexing activities; and table space problems. A database administrator can use this information to diagnose problems, tune the database, or simply monitor the database.

Administration notification log messages are also logged to the db2diag.log using a standardized message format.

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.

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 the GET DBM CFG command.

  Look for the following variable:

  ```
  Notify Level                              (NOTIFYLEVEL) = 3
  ```
- To alter the setting, use the UPDATE DBM CFG command. For example:

  ```
  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.

The Administration log, as with all logs in the database, grows continuously. Some logs grow more quickly than others depending on what is logged in each file. When a log gets too large, you should back it up and then erase it. A new log is generated automatically the next time it is 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.

```
2006-02-15-19.33.37.630000  1     Instance:DB2  2    Node:000  3
PID:940(db2syscs.exe) TID: 660 4    Appid:*LOCAL.DB2.020205091435  5
recovery manager  6   sqlpresr  7   Probe:1  8    Database:SAMPLE   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 database partition generating the message. (In a non-partitioned database, the value is ″000″.)
4. The process identifier (PID), followed by the name of the process, followed by the thread identifier (TID) that are responsible for the generation of the message.
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.DB2.020205091435.

   This value is the same as the *appl_id* monitor element data. For detailed information about how to interpret this value, see the documentation for the *appl_id* monitor element.

   To identify more about a particular application ID, either:

   - Use the LIST APPLICATIONS command on a DB2 server or LIST DCS APPLICATIONS on a DB2 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 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 component that is writing the message. For messages written by user applications using the `db2AdminMsgWrite` API, the function will read “User Function”.
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.

About DB2 diagnostic log (db2diag.log) files

The contents of the DB2 diagnostic log files including a description of the different levels of what is logged is presented along with directions on how to set the error capture level.

Setting the diagnostic log file error capture level

The DB2 diagnostic log is a file that contains text information logged by DB2. This information is used for troubleshooting and is primarily intended for DB2 customer support.

The information that DB2 records in the db2diag.log is determined by the DIAGLEVEL setting.

- To check the current setting, issue the command GET DBM CFG.

  Look for the following variable:

  ```
  Diagnostic error capture level              (DIAGLEVEL) = 3
  ```
- To change the value dynamically, use the UPDATE DBM CFG command.

  To change a database manager configuration parameter online:

  ```
  db2 attach to <instance-name>
  db2 update dbm cfg using <parameter-name> <value>
  db2 detach
  ```

  For example:

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

  where X is the desired notification level. If you are diagnosing a problem that can be reproduced, support personnel may suggest that you use DIAGLEVEL 4 while performing troubleshooting.

Interpreting the db2diag.log file informational record

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

An example of an informational record is as follows:

```
2006-02-09-18.07.31.059000-300 I1H917           LEVEL: Event
PID     : 3140                 TID  : 2864       PROC : db2start.exe
INSTANCE: DB2                  NODE : 000
FUNCTION: DB2 UDB, RAS/PD component, _pdlogInt, probe:120
START   : New Diagnostic Log file
DATA #1 : Build Level, 124 bytes
Instance "DB2" uses "32" bits and DB2 code release "SQL09010"
with level identifier "01010107".
Informational tokens are "DB2 v9.1.0.190", "s060121", "", Fix Pack "0".
DATA #2 : System Info, 1564 bytes
System: WIN32_NT MYSRVR Service Pack 2 5.1 x86 Family 15, model 2, stepping 4
CPU: total:1 online:1 Cores per socket:1 Threading degree per core:1
Physical Memory(MB): total:1024 free:617 available:617
Virtual  Memory(MB): total:2462 free:2830
Swap     Memory(MB): total:1438 free:2213
Information in this record is only valid at the time when this file was created
(see this record's time stamp)
```

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 might be useful.

Interpreting diagnostic log file entries

Use the db2diag.log analysis tool (db2diag) to filter and format the db2diag.log file. With the addition of administration notification log messages being logged to the db2diag.log using a standardized message format, viewing the db2diag.log first is a recommended choice to understand what has been happening to the database.

As an alternative to using db2diag, you can 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 and diagnostic logs grow *continuously*. When they get too large, back them up and then erase the files. 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.

```
2007-05-18-14.20.46.973000-240 1  I27204F655 2  LEVEL: Info 3
PID : 3228 4  TID : 8796 5  PROC : db2syscs.exe 6
INSTANCE: DB2MPP 7  NODE : 002 8  DB : WIN3DB1 9
APPHDL : 0-51 10  APPID: 9.26.54.62.45837.070518182042 11
AUTHID : UDBADM 12
EDUID : 8796 13  EDUNAME: db2agntp 14  (WIN3DB1) 2
FUNCTION: 15  DB2 UDB, data management, sqldInitDBCB, probe:4820
DATA #1 : 16  String, 26 bytes
Setting ADC Threshold to:
DATA #2 : unsigned integer, 8 bytes
1048576
```

Legend:

1. A timestamp and timezone for the message.

Note: Timestamps in the db2diag.log contain a time zone. For example: `2006-02-13-14.34.35.965000-300`, where "-300" is the difference between UTC (Coordinated Universal Time, formerly known as GMT) and local time at the application server in minutes. Thus -300 represents UTC - 5 hours, for example, EST (Eastern Standard Time).

2. The record ID field. The db2diag.log's recordID specifies the file offset at which the current message is being logged (for example, "27204") and the message length (for example, "655") for the platform where the DB2 diagnostic log was created.

3. The diagnostic level associated with an error message. For example, `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 database 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 9.26.54.62.45837.070518182042.

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.

This value is the same as the *appl_id* monitor element data. For detailed information about how to interpret this value, see the documentation for the *appl_id* monitor element.

To identify more about a particular application ID, either:

- Use the LIST APPLICATIONS command on a DB2 server or LIST DCS APPLICATIONS on a DB2 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 database partition name and its TCP/IP address.
- Use the GET SNAPSHOT FOR APPLICATION command to view a list of application IDs.
- Use the db2pd -applications -db <sample> command.

12 The authorization identifier.

13 The engine dispatchable unit identifier.

14 The name of the engine dispatchable unit.

15. The product name (″DB2″), component name (“data management”), and function name (“sqlInitDBCB”) that is writing the message (as well as the probe point (“4820”) within the function).

16. The information returned by a called function. There may be multiple data fields returned.

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>
AUTHID  : <authID>
EDUID   : <eduID>           EDUNAME: <engine dispatchable unit name>
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 ...
```

The fields which were not already explained in the example, are:

- `<source>` Indicates the origin of the logged error. (You can find it at the end of the first line in the sample.) The possible values are:
  - `origin` - message is logged by the function where error originated (inception point)
  - `OS` - error has been produced by the operating system
  - `received` - error has been received from another process (client/server)
  - `sent` - error has been sent to another process (client/server)
- `MESSAGE` Contains the message being logged. It consists of:
  - `<messageID>` - message number, for example, ECF=0x9000004A or DIA8604C
  - `<msgText>` - error description

  When the `CALLED` field is also present, `<msgText>` is an impact of the error returned by the `CALLED` function on the function logging a message (as specified in the FUNCTION field)
- `CALLED` This is the function that returned an error. It consists of:
  - `<prodName>` - The product name: ″OS″, ″DB2″, ″DB2 Tools″ or ″DB2 Common″
  - `<compName>` - The component name ('-' in case of a system call)
  - `<funcName>` - The called function name
- `OSERR` This is the operating system error returned by the CALLED system call. (You can find it at the end of the same line as CALLED.) It consists of:
  - `<errorName>` - the system specific error name
  - `<errno>` - the operating system error number
- `ARG` This section lists the arguments of a function call that returned an error. It consists of:
  - `<N>` - The position of an argument in a call to the ″called″ function
  - `<typeTitle>` - The label associated with the Nth argument typename
  - `<typeName>` - The name of the type of argument being logged
  - `<size>` - The size of argument to be logged
- `DATA` This contains any extra data dumped by the logging function. It consists of:
  - `<N>` - The sequential number of data object being dumped
  - `<typeTitle>` - The label of data being dumped
  - `<typeName>` - The name of the type of data field being logged, for example, PD_TYPE_UINT32, PD_TYPE_STRING
  - `<size>` - The size of a data object

db2cos (callout script) output files

A db2cos script is invoked by default when the database manager cannot continue processing due to a panic, trap, segmentation violation or exception. Each default db2cos script will invoke db2pd commands to collect information in an unlatched manner. The names for the db2cos scripts are in the form db2cos_hang, db2cos_trap, and so on. Each script behaves in a similar way except db2cos_hang which is called from the db2fodc tool.

The default db2cos scripts are found under the bin directory. On UNIX operating systems, this directory is read-only. You can copy the db2cos script file to the adm directory and modify the file at that location if you need to. If a db2cos script is found in the adm directory, it is run; otherwise, the script in the bin directory is run.

In a multiple partition configuration, the script will only be invoked for the trapping agent on the partition encountering the trap. If there is a need to collect information from other partitions, you can update the db2cos script to use the db2_all command or, if all of the partitions are on the same machine, specify the -alldbpartitionnums option on the db2pd command.

The types of signals that trigger the invocation of db2cos are also configurable via the db2pdcfg -cos command. The default configuration is for the db2cos script to run when either a panic or trap occurs. However, generated signals will not launch the db2cos script by default.

The order of events when a panic, trap, segmentation violation or exception occurs is as follows:

1. Trap file is created
2. Signal handler is called
3. db2cos script is called (depending on the db2cos settings enabled)
4. An entry is logged in the Administration Notification Log
5. An entry is logged in the db2diag.log

The default information collected by the db2pd command in the db2cos script includes details about the operating system, the Version and Service Level of the installed DB2 product, the database manager and database configuration, as well as information about the state of the agents, memory pools, memory sets, memory blocks, applications, utilities, transactions, buffer pools, locks, transaction logs, table spaces and containers. In addition, it provides information about the state of the dynamic, static, and catalog caches, table and index statistics, the recovery status, as well as the reoptimized SQL statements and an active statement list. If you need to collect further information, you simply update the db2cos script with the additional commands.

When the default db2cos script is called, it produces output files in the directory specified by the DIAGPATH database manager configuration parameter. The files are named XXX.YYY.ZZZ.cos.txt, where XXX is the process ID (PID), YYY is the thread ID (TID) and ZZZ is the database partition number (or 000 for single partition databases). If multiple threads trap, there will be a separate invocation of the db2cos script for each thread. In the event that a PID and TID combination occurs more than once, the data will be appended to the file. There will be a timestamp present so you can distinguish the iterations of output.

The db2cos output files will contain different information depending on the commands specified in the db2cos script. If the default script is not altered, entries similar to the following will appear (followed by detailed db2pd output):

```
2005-10-14-10.56.21.523659
PID     : 782348               TID  : 1           PROC : db2cos
INSTANCE: db2inst1             NODE : 0           DB   : SAMPLE
APPHDL  :                      APPID: *LOCAL.db2inst1.051014155507
FUNCTION: oper system services, sqloEDUCodeTrapHandler, probe:999
EVENT   : Invoking /home/db2inst1/sqllib/bin/db2cos from
oper system services sqloEDUCodeTrapHandler
Trap Caught

Instance db2inst1 uses 64 bits and DB2 code release SQL09010
...
Operating System Information:

OSName:   AIX
NodeName: n1
Version:  5
Release:  2
Machine:  000966594C00

...
```

The db2diag.log will contain entries related to the occurrence as well. For example:

```
2005-10-14-10.42.17.149512-300 I19441A349         LEVEL: Event
PID     : 782348               TID  : 1           PROC : db2sysc
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:10
START   : Invoking /home/db2inst1/sqllib/bin/db2cos from oper system
services sqloEDUCodeTrapHandler

2005-10-14-10.42.23.173872-300 I19791A310         LEVEL: Event
PID     : 782348               TID  : 1           PROC : db2sysc
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:20
STOP    : Completed invoking /home/db2inst1/sqllib/bin/db2cos

2005-10-14-10.42.23.519227-300 E20102A509         LEVEL: Severe
PID     : 782348               TID  : 1           PROC : db2sysc
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.

2005-10-14-10.42.23.520111-300 E20612A642         LEVEL: Severe
PID     : 782348               TID  : 1           PROC : db2sysc
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
0x0FFFFFFFFFFFD5C0 : 0000 000B 0000 0000 0000 0009 0000 0000    ................
0x0FFFFFFFFFFFD5D0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5E0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5F0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
```

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:

```
2007-05-18-12.28.11.277956-240 I24861950A192 LEVEL: Severe
PID:1056930 TID:225448 NODE:000 Title: dynamic memory buffer
Dump File:/home/svtdbm5/sqllib/db2dump/1056930.225448.000.dump.bin
```

Note: For partitioned database environments, 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/6881492.2.010.dump.bin
```

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 the ″function call stack″ or ″stack trace.″ The trap file also contains additional information about the state of the process when the signal or exception was caught.

The files are located in the directory specified by the DIAGPATH database manager configuration parameter.

On all platforms, the trap file name begins with a process identifier (PID), followed by a thread identifier (TID), followed by the partition number (000 on single partition databases), and concluded with ".trap.txt".

There are also diagnostic traps, generated by the code when certain conditions occur which don't warrant crashing the instance, but where it is useful to see the stack. Those traps are named with the PID in decimal format, followed by the partition number (0 in a single partition database).

Examples:

- 6881492.2.000.trap.txt is a trap file with a process identifier (PID) of 6881492, and a thread identifier (TID) of 2.
- 6881492.2.010.trap.txt is a trap file whose process and thread is running on partition 10.

You can generate trap files on demand using the db2pd command with the -stack all or -dump option. In general, though, this should only be done as requested by DB2 Support.

You can generate stack trace files with db2pd -stacks or db2pd -dumps commands. These files have the same contents as trap file but are generated for diagnostic purposes only. Their names will be similar to 6881492.2.000.stack.txt.

Formatting trap files (Windows)

A tool called db2xprt.exe is available to let you format trap files (*.TRP). It formats the DB2 database binary trap files into a human readable ASCII file.

The tool uses DB2 symbol files in order to format the trap files. A subset of these .PDB files are included with the DB2 database products.

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

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

Platform specific error log information

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 information made available in the DB2 files. The other files and utilities provide access to information contained in logs and traces that is concerned with the following areas:

- Operating systems
- Applications and third-party vendors
- Hardware

Based on your operating environment, there might be more places outside of what has been described here, so be aware of all of the potential areas you might 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 information into operating system error logs. However, sometimes additional information is required. In those cases, you need to identify what hardware diagnostic files and utilities might 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.

Some information, such as information from hardware logs, is time-sensitive. When an error occurs you should make every effort to gather as much information as you can from the relevant sources as soon as is possible.

In summary, to completely understand and evaluate a problem, you might 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 might help the investigation.

System core files (Linux and 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 diagpath by default unless otherwise configured using the values in the DB2FODC registry variable. Note that system core files are distinct from DB2 trap files.

Accessing system core file information (Linux and 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 is in error, or whether an operating system or application error is responsible for the problem.

- You must have the dbx command installed. This command is operating system-specific: on AIX and Solaris, use dbx; on HP-UX, use xdb, and on Linux use gdb.
- On AIX, ensure that the full core option has been enabled using the chdev command or smitty.

The following steps can be used to determine the function that caused the core file dump to occur.

1. Enter the following command from a UNIX command prompt:
   ```
   dbx program_name core_filename
   ```
   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. Examine the call stack in the core file. Information about how to do this can be obtained by issuing `man dbx` from a UNIX command prompt
3. To end the dbx command, type quit at the dbx prompt.

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". 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

View the event logs using the Windows Event Viewer. The method used to launch the viewer will differ, depending on the Windows operating system you are using.

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

Exporting event logs (Windows)

From the Windows event viewer, you can export event logs in two formats: log-file format and text or comma-delimited format.

Export the event logs from the Windows event viewer.

- The log-file format (*.evt) allows you to load the data back into an event viewer (for example on another machine). This format is easy to work with since you can use the viewer to switch the chronology order, filter for certain events, and advance forwards or backwards.
- The text or comma-delimited formats (*.txt and *.csv, respectively) allow you to open the logs in most text editors. They also avoid a potential problem with respect to 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 overlook key events because of time zone differences. Text files are also easier to search.

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.

Locate the Dr. Watson log file. The default path is <install_drive>:\Documents and Settings \All Users\Documents\DrWatson

Combining DB2 database and OS diagnostics

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 might 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
- 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 exercise 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. Start the DB2 instance 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"
     "db2cos9402136.0"
     "db2cos9402840.0"
     "db2dbamr.prf"
     "db2diag.bak"
     "db2eventlog.000"
     "db2misc.prf"
     "db2nodes.cfg"
     "db2profile.bat"
     "db2systm"
     "db2tools.prf"
     "HealthRulesV82.reg"
     "db2dasdiag.log"
     ...
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"
     "db2supp_system.zip"
     "dbm_detailed.supp_cfg"
     "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
   - Operating system level
   - User environment
   - User resource limits (UNIX ulimit command)

Exercise 2: Locating environment information in a DB2 trap file

1. Ensure a DB2 instance is started, then issue

```
db2pd -stack all
```

   The call stacks are placed in files in the diagnostic directory (as defined in the database manager configuration parameter DIAGPATH).

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 might not appear in all trap files)
   - Environment variables (this information might not appear in all trap files)
   - map output (shows loaded libraries)

Example trap file from Windows (truncated):

```
...
<DB2TrapFile version="1.0">
<Trap>
<Header>
DB2 build information: DB2 v9.1.0.190 s060121 SQL09010
timestamp: 2006-02-17-14.03.43.846000
uname: S:Windows
comment:
process id: 940
thread id: 3592
</Header>
<SystemInformation>
Number of Processors: 1
Processor Type: x86 Family 15 Model 2 Stepping 4
OS Version: Microsoft Windows XP, Service Pack 2 (5.1)
Current Build: 2600
</SystemInformation>
<MemoryInformation>
<Usage>
Physical Memory:    1023 total,     568 free.
Virtual Memory :    2047 total,    1882 free.
Paging File    :    2461 total,    2011 free.
Ext. Virtual   :       0 free.
</Usage>
</MemoryInformation>
<EnvironmentVariables>
<![CDATA[
[e] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2_EXTSECURITY=YES
[g] DB2SYSTEM=MYSRVR
[g] DB2PATH=C:\Program Files\IBM\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS00
]]></EnvironmentVariables>
```

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 you can correlate a reported system problem with DB2 errors, you 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 which can 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 can 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, or exceptions from other software that is failing)
- Kernel panics, out-of-filesystem-space, and out-of-swap-space errors (which can prevent the system from creating or forking a new process)

System logs can help to rule out crash entries in the db2diag.log as causes for concern. If you see crash recovery in DB2 Administration Notification or DB2 diagnostic logs with no preceding errors, the DB2 crash recovery is likely a result of a system shutdown.

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

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

Chapter 2. Troubleshooting DB2

In general, troubleshooting requires that you isolate and identify a problem, then seek a resolution. This section will provide troubleshooting information related to specific features of DB2 products.

As common problems are identified, the findings will be added to this section in the form of checklists. If the checklist does not lead you to a resolution, you can collect additional diagnostic data and analyze it yourself, or submit the data to IBM Software Support for analysis.

The following questions direct you to appropriate troubleshooting tasks:

1. Have you applied all known fix packs? If not, consider "Applying fix packs" in *Quick Beginnings for DB2 Servers*.
2. Does the problem occur when you are:
   - Installing DB2 database servers or clients? If so, see the topic "Collect data for installation problems" elsewhere in this book.
   - Creating, dropping, updating or migrating an instance or the DB2 Administration Server (DAS)? If so, see the topic "Collect data for DAS and instance management problems" elsewhere in this book..
   - Moving data using EXPORT, IMPORT, LOAD or db2move commands? If so, see the topic "Collect data for data movement problems" elsewhere in this book.

If your problem does not fall into one of these categories, basic diagnostic data might still be required if you are contacting IBM Software Support. You should .see the topic "Collect data for DB2" elsewhere in this book.

Current release troubleshooting guidance

Guidance is given regarding potential problems you may encounter when working with new and changed database functions and features. The recent changes to the database operations can result in you forgetting or overlooking the best possible way of taking advantage of the new or changed features. The following troubleshooting topics suggest possible problem areas and suggest directions for you so that you can take full advantage of the new and changed database functions and features introduced in this release.

Troubleshooting high availability

Tivoli System Automation for Multiplatforms (SA MP) Base Component is not installed by DB2 Version 9.5 GA on AIX 6.1

The IBM Tivoli® SA MP Base Component that is included in the DB2 Version 9.5 GA High Availability Feature does not support the AIX 6.1 operating system. To obtain the appropriate version of the SA MP Base Component for AIX 6.1, install DB2 Version 9.5 Fix Pack 1 or later fix packs.

Symptoms

If you install a DB2 Version 9.5 GA database product on AIX 6.1, the installer will detect that you are using AIX 6.1 and will not install the SA MP Base Component.

Causes

The SA MP Base Component that is bundled with DB2 Version 9.5 GA does not support AIX 6.1.

Resolving the problem

When you install DB2 Version 9.5 Fix Pack 1 or later fix packs on AIX 6.1, the SA MP Base Component will be installed successfully.

Troubleshooting installation

Errors when installing a DB2 database product to the default path on a system WPAR (AIX)

Various errors can occur if you install DB2 database products in the default installation path (/opt/IBM/db2/V9.5) on a system workload partition (WPAR) on AIX 6.1. To avoid these problems, install DB2 database products on a file system that is accessible only to the WPAR.

Symptoms

If you install DB2 database products in the /usr or /opt directories on a system WPAR, various errors can occur depending on how you configured the directories. System WPARs can be configured to either share the /usr and /opt directories with the global environment (in which case the /usr and /opt directories will be readable but not write accessible from the WPAR) or to have a local copy of the /usr and /opt directories.

In the first scenario, if a DB2 database product is installed to the default path on the global environment, that installation will be visible in the system WPAR. This will give the appearance that DB2 is installed on the WPAR, however attempts to create a DB2 instance will result in this error: `DBI1288E The execution of the program db2icrt failed. This program failed because you do not have write permission on the directory or file /opt/IBM/db2/V9.5/profiles.reg,/opt/IBM/db2/V9.5/default.env.`

In the second scenario, if a DB2 database product is installed to the default path on the global environment then when the WPAR creates the local copy of the /usr and /opt directories the DB2 database product installation will also be copied. This can cause unexpected problems if a system administrator attempts to use the database system. Since the DB2 database product was intended for another system, inaccurate information might be copied over. For example, any DB2 instances originally created on the global environment will appear to be present in the WPAR. This can cause confusion for the system administrator with respect to which instances are actually installed on the system.

Causes

These problems are caused by installing DB2 database products in /usr or /opt directories on a system WPAR.

Resolving the problem

Do not install DB2 database products in the default path on the global environment.

Mount a file system that is accessible only to the WPAR and install the DB2 database product on that file system.

Troubleshooting partitioned database environments

There are unique considerations to troubleshooting when you have an environment that has a partitioned database. Fast communication manager (FCM) will encounter problems if the host file has a specific entry that is valid when working in other than a partitioned database environment.

FCM problems related to 127.0.0.2 (Linux and UNIX)

In a partitioned database environment, the fast communications manager (FCM) might encounter problems if there is an entry for 127.0.0.2 in the /etc/hosts file.

Symptoms

Various error messages might occur, depending on the circumstances. For example, the following error can occur when you create a database: SQL1229N The current transaction has been rolled back because of a system error. SQLSTATE=40504

Causes

The problem is caused by the presence of an entry for the IP address 127.0.0.2 in the /etc/hosts file, where 127.0.0.2 maps to the fully qualified hostname of the machine. For example:

```
127.0.0.2 ServerA.ibm.com ServerA
```

where ″ServerA.ibm.com″ is the fully qualified hostname.

Environment

The problem is limited to DB2 Enterprise Server Edition with the DB2 Database Partitioning Feature.

Resolving the problem

Remove the entry from the /etc/hosts file, or convert it into a comment. For example:

```
# 127.0.0.2 ServerA.ibm.com ServerA
```

Creating a database partition on an encrypted file system (AIX)

AIX 6.1 supports the ability to encrypt a JFS2 file system or set of files. This feature is not supported with partitioned database environments in DB2 database products. An SQL10004C error will occur if you attempt to create a partitioned database environment using EFS (encrypted file systems) on AIX.

Symptoms

If you attempt to create a database on an encrypted file system in a multiple partition database environment, you will receive the following error: SQL10004C An I/O error occurred while accessing the database directory. SQLSTATE=58031

Causes

At this time it is not possible to create a partitioned database environment using EFS (encrypted file systems) on AIX. Since partitioned database partitions use rsh or ssh, the keystore in EFS is lost and database partitions are unable to access the database files that are stored on the encrypted file system.

Diagnosing the problem

The DB2 diagnostic log file (db2diag.log) will contain the error message and the following text: `OSERR   : ENOATTR (112) "No attribute found".`

Resolving the problem

To create a database successfully in a partitioned database environment, you must have a file system that is available to all of the machines and it must not be an encrypted file system.

Troubleshooting optimization guidelines and profiles

Diagnostics support for optimization guidelines (passed by optimization profiles) is provided via EXPLAIN tables.

You will receive an SQL0437W warning with reason code 13 if the optimizer does not apply an optimization guideline. Diagnostic information detailing why an optimization guideline was not applied is added to the EXPLAIN tables. There are two EXPLAIN tables for receiving optimizer diagnostic output:

- EXPLAIN_DIAGNOSTIC - Each entry in this table represents a diagnostic message pertaining to the optimization of a particular statement. Each diagnostic message is represented using a numeric code.
- EXPLAIN_DIAGNOSTIC_DATA - Each entry in this table is diagnostic data relating to a particular diagnostic message in the EXPLAIN_DIAGNOSTIC table.

The DDLs used to create the diagnostic explain tables is shown below in Figure 1 on page 31.

The following steps can help you troubleshoot problems that occur when you are using optimization guidelines:

1. "Verify that optimization guidelines have been used" in *Tuning Database Performance*.
2. Examine the full error message using the built-in "EXPLAIN_GET_MSGS table function" in *Administrative Routines and Views*.

If you complete these steps but cannot yet identify the source of the problem, begin collecting diagnostic data and consider contacting IBM Software Support.

```
CREATE TABLE EXPLAIN_DIAGNOSTIC
      ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
        EXPLAIN_TIME      TIMESTAMP    NOT NULL,
        SOURCE_NAME       VARCHAR(128) NOT NULL,
        SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
        SOURCE_VERSION    VARCHAR(64)  NOT NULL,
        EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
        STMTNO            INTEGER      NOT NULL,
        SECTNO            INTEGER      NOT NULL,
        DIAGNOSTIC_ID     INTEGER      NOT NULL,
        CODE              INTEGER      NOT NULL,
        PRIMARY KEY (EXPLAIN_REQUESTER,
                     EXPLAIN_TIME,
                     SOURCE_NAME,
                     SOURCE_SCHEMA,
                     SOURCE_VERSION,
                     EXPLAIN_LEVEL,
                     STMTNO,
                     SECTNO,
                     DIAGNOSTIC_ID),
        FOREIGN KEY (EXPLAIN_REQUESTER,
                     EXPLAIN_TIME,
                     SOURCE_NAME,
                     SOURCE_SCHEMA,
                     SOURCE_VERSION,
                     EXPLAIN_LEVEL,
                     STMTNO,
                     SECTNO)
        REFERENCES EXPLAIN_STATEMENT ON DELETE CASCADE);

   CREATE TABLE EXPLAIN_DIAGNOSTIC_DATA
      ( EXPLAIN_REQUESTER VARCHAR(128) NOT NULL,
        EXPLAIN_TIME      TIMESTAMP    NOT NULL,
        SOURCE_NAME       VARCHAR(128) NOT NULL,
        SOURCE_SCHEMA     VARCHAR(128) NOT NULL,
        SOURCE_VERSION    VARCHAR(64)  NOT NULL,
        EXPLAIN_LEVEL     CHAR(1)      NOT NULL,
        STMTNO            INTEGER      NOT NULL,
        SECTNO            INTEGER      NOT NULL,
        DIAGNOSTIC_ID     INTEGER      NOT NULL,
        ORDINAL           INTEGER      NOT NULL,
        TOKEN             VARCHAR(1000),
        TOKEN_LONG        BLOB(3M) NOT LOGGED,
        FOREIGN KEY (EXPLAIN_REQUESTER,
                     EXPLAIN_TIME,
                     SOURCE_NAME,
                     SOURCE_SCHEMA,
                     SOURCE_VERSION,
                     EXPLAIN_LEVEL,
                     STMTNO,
                     SECTNO,
                     DIAGNOSTIC_ID)
        REFERENCES EXPLAIN_DIAGNOSTIC ON DELETE CASCADE);
```

Note: The EXPLAIN_REQUESTOR, EXPLAIN_TIME, SOURCE_NAME, SOURCE_SCHEMA, SOURCE_VERSION, EXPLAIN_LEVEL, STMTNO, and SECTNO columns are part of both tables in order to form the foreign key to the EXPLAIN_STATEMENT table and the parent-child relationship between EXPLAIN_DIAGNOSTIC and EXPLAIN_DIAGNOSTIC_DATA.

Figure 1. DDLs used to create the diagnostic explain tables

This DDL is included in the EXPLAIN.DDL file located in the misc subdirectory of the sqllib directory.

Troubleshooting the storage key support

Storage protection keys, hardware keys at a thread level, are used to help the DB2 engine have greater resilience by protecting memory from invalid access attempts. Follow the steps below for any error that you have encountered while enabling this feature or any trap that you have encountered while this feature is enabled.

Diagnosing registry variable errors

When setting the registry variables “DB2_MEMORY_PROTECT” and “DB2_THREAD_SUSPENSION” in *Data Servers, Databases, and Database Objects Guide* , an `Invalid value` (DBI1301E) error was returned. This error occurs for one of the following reasons:

- The value given for the registry variable is invalid. Refer to the registry variable usage for the applicable variable, DB2_MEMORY_PROTECT or DB2_THREAD_SUSPENSION.
- If the error occurs when you are setting the DB2_MEMORY_PROTECT variable, then the hardware and operating system may not support storage protection keys and the feature cannot be enabled. Storage protection keys are available in POWER6 processors and are supported as of the AIX 5L Version 5.3, with the 5300-06 Technology Level, operating system.
- If the error occurs when you are setting the DB2_THREAD_SUSPENSION variable to ON, then the DB2_MEMORY_PROTECT variable was not set prior to setting the DB2_THREAD_SUSPENSION variable. Use the **db2set** command to set the DB2_MEMORY_PROTECT variable to YES. Next, use the **db2set** command to set the DB2_THREAD_SUSPENSION variable to ON. Finally, stop and restart your DB2 instance to activate the registry variable changes.

Diagnosing traps

The DB2 instance prepares the first occurrence data capture (FODC) package for the trap that you have encountered. If the DB2 instance has been configured for greater database resiliency, the DB2_MEMORY_PROTECT variable is set to YES and the DB2_THREAD_SUSPENSION variable is set to ON, then the DB2 instance has also determined whether or not the trap is sustainable. The term ’sustainable’ means that the trapped DB2 engine thread has been suspended or terminated and the DB2 instance continues to run. Perform these steps:

1. Use a text editor to view the administration notification log file. You will see error message “ADM14010C” from the *Message Reference, Volume 1* if the trap was sustainable and the DB2 instance is still running. Otherwise, you will see error message “ADM14011C” from the*Message Reference, Volume 1* and the DB2 instance has shutdown.
2. Note the directory name of the FODC information as specified in the appropriate error message above.
3. If the trap was sustained, stop the DB2 instance at your earliest convenience. Since a DB2 engine thread is suspended when the trap is sustained, the **db2stop** and STOP DATABASE MANAGER commands will hang if they are used to stop the DB2 instance. Instead, you must use the **db2_kill** command to stop the DB2 instance and remove the suspended DB2 engine thread.
4. Restart the DB2 instance using the **db2start** or START DATABASE MANAGER command.
5. Contact IBM Customer Support and refer to the FODC diagnostic information to resolve the cause of the trap.

Data compression dictionary is not automatically created

You have a large table but no data compression dictionary is created. You would like to understand why the creation of the data compression dictionary did not occur as you were expecting.

You may find yourself in the following situation:

- You have a table where the COMPRESS attribute has been set to YES.
- The table has existed for some time and data has been added and removed.
- The size of the table appears to be close to the threshold size (approximately 1 to 2 MB). You are expecting the data compression dictionary to be automatically created.
- You run a table data population operation (such as INSERT, LOAD INSERT, or REDISTRIBUTE) which you expect will increase the size of the table beyond the threshold size.
- Automatic creation of the data compression dictionary does not occur. The data compression dictionary is not created and placed into the table. You expect compression to occur on data added to the table after that point, but the data remains decompressed.

Why is the table data compression not occurring?

Although the table size is larger than the threshold size to allow automatic creation of the data compression dictionary, there is another condition that is checked. The condition is that there must be sufficient data present in the table to allow creation of the dictionary. Past activity against the data in the table may also have included the deletion or removal of data. There may be large sections within the table where there is no data. This is how you can have a large table which meets or exceeds the table size threshold, but there may not be enough data in the table to allow the creation of the dictionary.

If you experience a lot of activity against the table, you need to reorganized the table on a regular basis. If you do not, the table size may be large, but it may be sparsely populated with data. Reorganizing the table will eliminate fragmented data and compact the data in the table. Following the reorganization, the table will be smaller and be more densely populated. The reorganized table will more accurately represent the amount of data in the table and may be smaller than the threshold size to allow automatic creation of the data compression dictionary.

Use the REORGCHK command to determine if a table needs to be reorganized.

Troubleshooting global variable problems

In general, troubleshooting applications with regard to global variables is not a problem if the user experiencing the problem has permission to READ the global variables. Having READ permission is all that is needed to know what the value of the global variable is by issuing a VALUES(Global Variable Name) statement. There will be cases where the user running the application will not have access to READ the global variable.

The first scenario illustrates a possible problem when referencing global variables that has a simple solution. The second scenario presents a more likely situation where the permission to READ the global variables needs to be granted to the appropriate users.

Scenario 1

References to global variables must be properly qualified. It is possible that there exists a variable with the same name and a different schema where the incorrect schema is encountered earlier in the PATH register value. One solution is to ensure that the references to the global variable are fully qualified.

Scenario 2

An application developer (developerUser) creates a highly complex series of procedures, views, triggers, and so on based upon the value of some global variables to which only he has read access. An end user of the application (finalUser) logs in and starts issuing SQL using the environment created by developerUser. finalUser complains to developerUser that he cannot see data that he should be allowed to see. As part of troubleshooting this problem, developerUser changes his authorization ID to that of finalUser, logs in as finalUser, and tries the same SQL as finalUser. developerUser finds that finalUser is right, and there is a problem.

developerUser has to determine whether finalUser sees the same global variable values as he does. developerUser runs SET SESSION USER to see the global variable values that the finalUser sees. Here is a proposed method to determine this problem and solve it.

developerUser asks the security administrator (secadmUser) to grant him permission to use SET SESSION USER as finalUser. Then developerUser logs in as himself and uses the SET SESSION AUTHORIZATION statement to set the SESSION_USER special register to that of finalUser. After running the SQL that is the problem, he then switches back to developerUser using another SET SESSION AUTHORIZATION statement. developerUser can now issue a VALUES statement and see the actual value of the global variable.

What follows is sample SQL showing the actions taken in the database by developerUser.

```
########################################################################
# developerUser connects to database and creates needed objects
########################################################################

db2 "connect to sample user developerUser using xxxxxxxx"

db2 "create table security.users \
(userid varchar(10) not null primary key, \
firstname varchar(10), \
lastname varchar(10), \
authlevel int)"

db2 "insert into security.users values ('ZUBIRI', 'Adriana', 'Zubiri', 1)"
db2 "insert into security.users values ('SMITH', 'Mary', 'Smith', 2)"
db2 "insert into security.users values ('NEWTON', 'John', 'Newton', 3)"

db2 "create variable security.gv_user varchar(10) default (SESSION_USER)"
db2 "create variable security.authorization int default 0"

# Create a procedure that depends on a global variable
db2 "CREATE PROCEDURE SECURITY.GET_AUTHORIZATION() \
SPECIFIC GET_AUTHORIZATION \
RESULT SETS 1 \
LANGUAGE SQL \
   SELECT authlevel INTO security.authorization \
   FROM security.users \
   WHERE userid = security.gv_user"

db2 "grant all on variable security.authorization to public"
db2 "grant execute on procedure security.get_authorization to public"
db2 "terminate"

########################################################################
# secadmUser grants setsessionuser
########################################################################
db2 "connect to sample user secadmUser using xxxxxxxx"
db2 "grant setsessionuser on user finalUser to user developerUser"
db2 "terminate"

########################################################################
# developerUser will debug the problem now
########################################################################

echo "------------------------------------------------------------"
echo " Connect as developerUser "
echo "------------------------------------------------------------"
db2 "connect to sample user developerUser using xxxxxxxx"

echo "------------------------------------------------------------"
echo " SET SESSION AUTHORIZATION = finalUser "
echo "------------------------------------------------------------"
db2 "set session authorization = finalUser"

echo "--- TRY to get the value of gv_user as finalUser (we should not be able to)"
db2 "values(security.gv_user)"

echo "--- Now call the procedure---"
db2 "call security.get_authorization()"

echo "--- if it works it should return 3 ---"
db2 "values(security.authorization)"

echo "------------------------------------------------------------"
echo " SET SESSION AUTHORIZATION = developerUser "
echo "------------------------------------------------------------"

db2 "set session authorization = developerUser"

echo "--- See what the variable looks like ----"
db2 "values(security.gv_user)"

db2 "terminate"
```

Troubleshooting work load management

A workload is a database object consisting of user-defined criteria that groups one or more units of work (UOW) within a database. A workload occurrence consists of one or more UOWs within a database connection that is associated with a workload. If the same workload is associated with another set of UOWs on a different database connection, then that set of UOWs is considered as a different workload occurrence.

There can be more than one workload occurrence running on the system concurrently for each workload.

When there are problems or difficulties with a workload or workload occurrence, there are types of information about each that you want to know. The types of information include:

- List of workload occurrences. Use the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURENCES table function. The output from this function includes system-wide unique identifiers for the application, and the unique UOW identifier.

- Identifier for the workload and workload occurrence. Use the WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function. The output from this function includes the unique UOW identifier and the unique activity identifier within a UOW.
- List of all activities (and requests) that are currently running under a workload occurrence. Use the WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function. The output from this function includes the unique UOW identifier and the unique activity identifier within a UOW.
- Content about the workload information found in memory at the time. Use the db2pd -workloads command. The output from this command includes the workload list, workload authority information, and a counter for the number of concurrent workload occurrences for each workload.

In addition, should you need to contact IBM during your troubleshooting of work load management problems, you will need to gather information about the db2agent associated with the situation. Use db2pd -stack to gather the needed information. After contacting IBM, you may be asked to perform a DB2 trace (db2trc).

Troubleshooting scripts

You may have internal tools or scripts that are based on the processes running in the database engine. These tools or scripts may no longer work because all agents, prefetchers, and page cleaners are now considered threads in a single, multi-threaded process.

Your internal tools and scripts will have to be modified to account for a threaded process. For example, you may have scripts that invoke the ps command to list the process names; and then perform tasks against certain agent processes. Your scripts will need to be rewritten.

The problem determination database command db2pd will have a new option -edu (short for “engine dispatchable unit”) to list all agent names along with their thread IDs. The db2pd -stack command continues to work with the threaded engine to dump individual EDU stacks or to dump all EDU stacks for the current node.

Troubleshooting data inconsistencies

Diagnosing where data inconsistencies exist within the database is very important. One way to determine data inconsistencies is to use the output from the INSPECT command to identify where a problem exists. When inconsistencies are found, you will have to decide how to deal with the problem.

Once you have determined that there is a data consistency problem, you have two options:

- Contact DB2 Service and ask for their assistance in recovering from the data inconsistency
- Drop and rebuild the database object that has the data consistency problem.

You will use the INSPECT CHECK variation from the INSPECT command to check the database, table space, or table that has evidence of a data inconsistency. Once the results of the INSPECT CHECK command are produced, you should format the inspection results using the db2inspf command.

If the INSPECT command does not complete then contact DB2 Service.

Troubleshooting index to data inconsistencies

Indexes must be accurate to allow quick access to the right data in tables otherwise your database is corrupt.

You can use the INSPECT command to carry out an online check for index to data inconsistency by using the INDEXDATA option in the cross object checking clause. Index data checking is not performed by default when using the INSPECT command; it must be explicitly requested.

When there is an error discovered due to an index data inconsistency while INSPECT performs an INDEXDATA inspection, the error message SQL1141N is returned. At the same time this error message is returned, data diagnostic information is collected and dumped to the db2diag.log. An urgent message is also logged in the administration notification log. Use the db2diag.log analysis tool to filter and format the contents of the db2diag. log file.

Locking implications

While checking for index to data inconsistencies by using the INSPECT command with the INDEXDATA option, the inspected tables are only locked in IS mode.

When the INDEXDATA option is specified, by default only the values of explicitly specified level clause options are used. For any level clause options which are not explicitly specified, the default levels (INDEX NORMAL and DATA NORMAL) are overwritten from NORMAL to NONE.

Collecting data for DB2

Sometimes you cannot solve a problem simply by troubleshooting the symptoms. In such cases, you need to collect diagnostic data. The diagnostic data that you need to collect and the sources from which you collect that data are dependent on the type of problem that you are investigating. These steps represent how to collect the base set of information that you typically need to provide when you submit a problem to IBM Software Support.

To obtain the most complete output, the db2support utility should be invoked by the instance owner.

To collect the base set of diagnostic information in a compressed file archive, enter the db2support command:

```
db2support <output_directory> -s -d <database_name> -c
```

Using -s will give system details about the hardware used and the operating system. Using -d will give details about the specified database. Using -c allows for an attempt to connect to the specified database.

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.

For specific symptoms, or for problems in a specific part of the product, you might need to collect additional data. Refer to the problem-specific ″Collecting data″ documents.

You can do any of the following tasks next:
- Analyze the data
- Submit the data to IBM Software Support

Collecting data for installation problems

If you are experiencing installation problems and cannot determine the cause of the problem, collect diagnostic data that either you or IBM Software Support can use to diagnose and resolve the problem.

To collect diagnostic data for installation problems:
1. Optional: Repeat the installation attempt with tracing enabled. For example:
   ```
   db2setup -t trace.out
   OR
   setup -t trace.out
   ```
2. Locate the installation log files.
   - On Windows, the default file name is "DB2-*ProductAbbreviation-DateTime*.log". For example: DB2-ESE-Wed Jun 21 11_59_37 2006.log. The default location for the installation log is the "My Documents"\DB2LOG\ directory.
   - On Linux and UNIX, the default file names are db2setup.log, db2setup.his, and db2setup.err.

     If you recreated the problem with tracing (or debug mode) enabled, additional files might be created, such as: dascrt.log, dasdrop.log, dasupdt.log, db2icrt.log.PID, db2idrop.log.PID, db2imigr.log.PID, and db2iupdt.log.PID, where PID is the process ID.

     The default location for all of these files is the /tmp directory.
3. Optional: If you intend to submit the data to IBM Software Support, collect data for DB2 as well.

Collecting data for data movement problems

If you are experiencing problems while performing data movement commands and you cannot determine the cause of the problem, collect diagnostic data that either you or IBM Software Support can use to diagnose and resolve the problem.
- To collect data for problems related to the db2move command, go to the directory where you issued the command. Locate the following file(s), depending on the action you specified in the command:
  - For the COPY action, look for files called COPY.*timestamp*.ERR and COPYSCHEMA.*timestamp*.MSG. If you also specified either LOAD_ONLY or DDL_AND_LOAD mode, look for a file called LOADTABLE.*timestamp*.MSG as well.
  - For the EXPORT action, look for a file called EXPORT.out.
  - For the IMPORT action, look for a file called IMPORT.out.
  - For the LOAD action, look for a file called LOAD.out.
- To collect data for problems related to EXPORT or IMPORT or LOAD commands, determine whether your command included the MESSAGES parameter. If it did, collect the output file. These utilities use the current directory and the default drive as the destination if you do not specify otherwise.
- To collect data for problems related to a REDISTRIBUTE command, look for a file called "*databasename.database_partition_groupname. timestamp*" on Linux and

UNIX and "*databasename. database_partition_groupname.date.time*" on Windows. It is located in *$HOME*/sqllib/db2dump directory or $DB2PATH\sqllib\redist respectively, where *$HOME* is the home directory of the instance owner.

Collecting data for DAS and instance management problems

If you are experiencing problems while performing DB2 Administration Server (DAS) or instance management and you cannot determine the cause of the problem, collect diagnostic data that either you or IBM Software Support can use to diagnose and resolve the problem.

These steps are only for situations where you can recreate the problem and you are using DB2 on Linux or UNIX.

To collect diagnostic data for DAS or instance management problems:

1. Repeat the failing command with tracing or debug mode enabled. Example commands:

   ```
   db2setup -t trace.out
   dascrt -u DASUSER -d
   dasdrop -d
   dasmigr -d
   dasupdt -d
   db2icrt -d INSTNAME
   db2idrop INSTNAME -d
   db2imigr -d INSTNAME
   db2iupdt -d INSTNAME
   ```
2. Locate the diagnostic files. More than one file might be present, so compare the timestamps to ensure that you are obtaining all of the appropriate files.

   The output will appear in the /tmp directory by default.

   Example file names are: dascrt.log, dasdrop.log , dasupdt.log , db2icrt.log.PID, db2idrop.log.PID, db2imigr.log.PID, and db2iupdt.log.PID, where PID is the process ID.
3. Provide the diagnostic file(s) to IBM Software Support.

If the problem is that the db2start or START DATABASE MANAGER command is failing, look for a file named db2start.*timestamp*.log in the insthome/sqllib/log directory, where insthome is the home directory for the instance owner. Likewise if the problem is that the db2stop or STOP DATABASE MANAGER command is failing, look for a file named db2stop.*timestamp*.log. These files will only appear if the database manager did not respond to the command within the amount of time specified in the start_stop_time database manager configuration parameter.

Analyzing data for DB2

After you collect data, you need to determine how that data can help you to resolve your particular problem. The type of analysis depends on the type of problem that you are investigating and the data that you have collected. These steps represent how to start your investigation of any basic DB2 diagnostic data.

To analyze diagnostic data, take the following actions:

- Have a clear understanding of how the various pieces of data relate to each other. For example, if the data spans more than one system, keep your data well organized so that you know which pieces of data come from which sources.
- Confirm that each piece of diagnostic data is relevant to the timing of the problem by checking timestamps. Note that data from different sources can have

different timestamp formats; be sure to understand the sequence of the different elements in each timestamp format so that you can tell when the different events occurred.

- Determine which data sources are most likely to contain information about the problem, and start your analysis there. For example, if the problem is related to installation, start your analysis with the installation log files (if any), rather than starting with the general product or operating system log files.
- The specific method of analysis is unique to each data source, but one tip that is applicable to most traces and log files is to start by identifying the point in the data where the problem occurs. After you identify that point, you can work backward in time through the data in order to unravel the root cause of the problem.
- If you are investigating a problem for which you have comparative data from an environment that is working and one that is not, start by comparing the operating system and product configuration details for each environment.

Analyzing data for installation problems

After you collect diagnostic data about installation problems, you can analyze the data to determine the cause of the problem. These steps are optional. If the cause of the problem is not easily determined, submit the data to IBM Software Support.

These steps assume that you have obtained the files described in Collecting data for installation problems.

1. Ensure that you are looking at the appropriate installation log file. Check the file's creation date, or the timestamp included in the file name (on Windows operating systems).
2. Determine whether the installation completed successfully.
   - On Windows operating systems, success is indicated by a message similar to the following at the bottom of the installation log file:
     ```
     Property(C): INSTALL_RESULT = Setup Complete Successfully
     === Logging stopped: 6/21/2006  16:03:09 ===
     MSI (c) (34:38) [16:03:09:109]:
     Product: DB2 Enterprise Server Edition - DB2COPY1 -- Installation operation
     completed successfully.
     ```
   - On Linux and UNIX operating systems, success is indicated by a message at the bottom of the installation log file (the one named db2setup.log by default).
3. OPTIONAL: Determine whether any errors occurred. If the installation completed successfully, but you received an error message during the installation process, locate these errors in the installation log file.
   - On Windows operating systems, most errors will be prefaced with "ERROR:" or "WARNING:". For example:
     ```
     1: ERROR:An error occurred while running the command
     "D:\IBM\SQLLIB\bin\db2.exe
     CREATE TOOLS CATALOG SYSTOOLS USE EXISTING DATABASE TOOLSDB FORCE" to
     initialize and/or migrate the DB2 tools catalog database.
     The return value is "4".

     1: WARNING:A minor error occurred while installing "DB2 Enterprise Server
     Edition - DB2COPY1" on this computer. Some features may not function
     correctly.
     ```
   - On Linux and UNIX operating systems, a file with a default name of db2setup.err will be present if any errors were returned by Java™ (for example, exceptions and trap information).

If you had enabled an installation trace, there will be more entries in the installation log files and the entries will be more detailed.

If analyzing this data does not help you to resolve your problem, and if you have a maintenance contract with IBM, you can open a problem report. IBM Software Support will ask you to submit any data that you have collected, and they might also ask you about any analysis that you performed.

If your investigation has not solved the problem, submit the data to IBM Software Support.

Submitting data to IBM Software Support

The steps assume that you have already opened a problem management record (PMR) with IBM Software Support.

You can send diagnostic data, such as log files and configuration files, to IBM Software Support using one of the following methods:

- FTP
- Electronic Service Request (ESR) tool
- To submit files (via FTP) to the Enhanced Centralized Client Data Repository (EcuRep):
  1. Package the data files that you collected into ZIP or TAR format, and name the package according to your Problem Management Record (PMR) identifier.

     Your file must use the following naming convention in order to be correctly associated with the PMR: xxxxx.bbb.ccc.yyy.yyy, where xxxxx is the PMR number, bbb is the PMR's branch number, ccc is the PMR's territory code, and yyy.yyy is the file name.
  2. Using an FTP utility, connect to the server ftp.emea.ibm.com.
  3. Log in as the userid ″anonymous″ and enter your e-mail address as your password.
  4. Go to the toibm directory. For example, cd toibm.
  5. Go to one of the operating system-specific subdirectories. For example, the subdirectories include: aix, linux, unix, or windows.
  6. Change to binary mode. For example, enter bin at the command prompt.
  7. Put your file on the server by using the put command. Use the following file naming convention to name your file and put it on the server. Your PMR will be updated to list where the files are stored using the format: xxxx.bbb.ccc.yyy.yyy. (xxx is the PMR number, bbb is the branch, ccc is the territory code, and yyy.yyy is the description of the file type such as tar.Z or xyz.zip.) You can send files to the FTP server, but you cannot update them. Any time that you need to subsequently change the file, you need to create a new file name.
  8. Enter the quit command.
- To submit files using the ESR tool:
  1. Sign onto ESR.
  2. On the Welcome page, enter your PMR number in the **Enter a report number** field, and click **Go**.
  3. Scroll down to the **Attach Relevant File** field.
  4. Click **Browse** to locate the log, trace, or other diagnostic file that you want to submit to IBM Software Support.

5. Click **Submit**. Your file is transferred to IBM Software Support through FTP, and it is associated with your PMR.

For more information about the EcuRep service, see IBM EMEA Centralized Customer Data Store Service.

For more information about ESR, see Electronic Service Request (ESR) help.

Chapter 3. Troubleshooting DB2 Connect

The system environment that uses DB2 Connect has unique possibilities for problems beyond those that would normally be involved with a local database. Gathering information about problems also has to involve communication protocols and hardware. In addition, remote databases are possible sources of problems and need to be considered.

Troubleshooting

The DB2 Connect environment involves multiple software, hardware and communications products. Troubleshooting is best approached by a process of elimination and refinement of the available data to arrive at a conclusion (the location of the error).

After gathering the relevant information and based on your selection of the applicable topic, proceed to the referenced section.

Gathering relevant information

Troubleshooting 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.

- Has the initial connection been successful?
- Is the hardware functioning properly?
- Are the communication paths operational?
- Have there been any communication network changes that would make previous directory entries invalid?
- Has the database been started?
- Is the communication breakdown between one or more clients and the DB2 Connect Server (gateway); between the DB2 Connect gateway and the host or System i™ database server; or between theDB2 Connect Personal Edition and the host or System i database server?
- What can you determine by the content of the message and the tokens returned in the message?
- Will using diagnostic tools such as db2trc, db2pd, or db2support 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?

Initial connection is not successful

Review the following questions and ensure that the installation steps were followed:

1. *Did the installation processing complete successfully?*
   - Were all the prerequisite software products available?
   - Were the memory and disk space adequate?
   - Was remote client support installed?

- Was the installation of the communications software completed without any error conditions?

2. *For UNIX operating systems, was an instance of the product created?*
   - As root did you create a user and a group to become the instance owner and sysadm group?
3. *If applicable, was the license information processed successfully?*
   - For UNIX operating systems, did you edit the nodelock file and enter the password that IBM supplied?
4. *Were the host or System i database server and workstation communications configured properly?*
   - There are three configurations that must be considered:
     a. The host or System i database server configuration identifies the application requester to the server. The host or System i server database management system will have system catalog entries that will define the requestor in terms of location, network protocol and security.
     b. The DB2 Connect workstation configuration defines the client population to the server and the host or System i server to the client.
     c. The client workstation configuration must have the name of the workstation and the communications protocol defined.
   - Problem analysis for not making an initial connection includes verifying that PU (physical unit) names are complete and correct, or verifying for TCP/IP connections that the correct port number and hostname have been specified.
   - Both the host or System i server database administrator and the Network administrators have utilities available to diagnose problems.
5. *Do you have the level of authority required by the host or System i server database management system to use the host or System i server database?*
   - Consider the access authority of the user, rules for table qualifiers, the anticipated results.
6. *If you attempt to use the Command Line Processor (CLP) to issue SQL statements against a host or System i database server, are you unsuccessful?*
   - Did you follow the procedure to bind the CLP to the host or System i database server?

Problems encountered after an initial connection

The following questions are offered as a starting point to assist in narrowing the scope of the problem.

1. *Are there any special or unusual operating circumstances?*
   - Is this a new application?
   - Are new procedures being used?
   - Are there recent changes that might be affecting the system? For example, have any of the software products or applications been changed since the application or scenario last ran successfully?
   - For application programs, what application programming interface (API) was used to create the program?
   - Have other applications that use the software or communication APIs been run on the user's system?
   - Has a fix pack recently been installed? If the problem occurred when a user tried to use a feature that had not been used (or loaded) on their operating system since it was installed, determine IBM's most recent fix pack and load it *after* installing the feature.

2. *Has this error occurred before?*
   - Are there any documented resolutions to previous error conditions?
   - Who were the participants and can they provide insight into a possible course of action?
3. *Have you explored using communications software commands that return information about the network?*
   - TCP/IP might have valuable information retrieved from using TCP/IP commands and daemons.
4. *Is there information returned in the SQLCA (SQL communication area) that can be helpful?*
   - Problem handling procedures should include steps to examine the contents of the SQLCODE and SQLSTATE fields.
   - SQLSTATEs allow application programmers to test for classes of errors that are common to the DB2 family of database products. In a distributed relational database network this field might provide a common base.
5. *Was DB2START executed at the Server?* Additionally, ensure that the DB2COMM environment variable is set correctly for clients accessing the server remotely.
6. *Are other machines performing the same task able to connect to the server successfully?* The maximum number of clients attempting to connect to the server might have been reached. If another client disconnects from the server, is the client who was previously unable to connect, now able to connect?
7. *Does the machine have the proper addressing?* Verify that the machine is unique in the network.
8. *When connecting remotely, has the proper authority been granted to the client?* Connection to the instance might be successful, but the authorization might not have been granted at the database or table level.
9. *Is this the first machine to connect to a remote database?* In distributed environments routers or bridges between networks might block communication between the client and the server. For example, when using TCP/IP, ensure that you can PING the remote host.

Diagnostic tools

When you encounter a problem, you can use the following:

- All diagnostic data including dump files, trap files, error logs, notification files, and alert logs are found in the path specified by the diagnostic data directory path (**diagpath**) database manager configuration parameter:

  If the value for this configuration parameter is null, the diagnostic data is written to one of the following directories or folders:
  – For Linux and UNIX environments: INSTHOME/sqllib/db2dump, where *INSTHOME* is the home directory of the instance.
  – For supported Windows environments:
    - If the **DB2INSTPROF** environment variable is not set then x:\SQLLIB\DB2INSTANCE is used where x:\SQLLIB is the drive reference and the directory specified in the **DB2PATH** registry variable, and the value of **DB2INSTANCE** has the name of the instance.

      Note: The directory does not have to be named SQLLIB.
    - If the **DB2INSTPROF** environment variable is set then x:\DB2INSTPROF\DB2INSTANCE is used where **DB2INSTPROF** is the

name of the instance profile directory and **DB2INSTANCE** is the name of the instance (by default, the value of **DB2INSTDEF** on Windows 32-bit operating systems).

- For Windows operating systems, you can use the Event Viewer to view the administration notification log.
- The available diagnostic tools that can be used include **db2trc**, **db2pd** and **db2support**.
- For Linux and UNIX operating systems, the **ps** command, which returns process status information about active processes to standard output.
- For UNIX operating systems, the core file that is created in the current directory when severe errors occur. It contains a memory image of the terminated process, and can be used to determine what function caused the error.

Common DB2 Connect problems

This topic lists the most common symptoms of connection problems encountered when using DB2 Connect. In each case, you are provided with:

- A combination of a message number and a return code (or protocol specific return code) associated with that message. Each message and return code combination has a separate heading, and the headings are ordered by message number, and then by return code.
- A symptom, usually in the form of a sample message listing.
- A suggested solution, indicating the probable cause of the error. In some cases, more than one suggested solution might be provided.

SQL0965 or SQL0969

Symptom

Messages SQL0965 and SQL0969 can be issued with a number of different return codes from DB2 for i5/OS, DB2 for z/OS®, and DB2 for VM & VSE.

When you encounter either message, you should look up the original SQL code in the documentation for the database server product issuing the message.

Solution

The SQL code received from the host or i5/OS database cannot be translated. Correct the problem, based on the error code, then resubmit the failing command.

SQL5043N

Symptom

Support for one or more communications protocols failed to start successfully. However, core database manager functionality started successfully.

Perhaps the TCP/IP protocol is not started on the DB2 Connect server. There might have been a successful client connection previously.

If `diaglevel = 4`, then `db2diag.log` might contain a similar entry, for example:

```
2001-05-30-14.09.55.321092   Instance:svtdbm5   Node:000
PID:10296(db2tcpcm)   Appid:none
common_communication  sqlcctcpconnmgr_child   Probe:46
DIA3205E Socket address "30090" configured in the TCP/IP
services file and
required by the TCP/IP server support is being used by another
process.
```

Solution

This warning is a symptom which signals that DB2 Connect, acting as a server for remote clients, is having trouble handling one or more client communication protocols. These protocols can be TCP/IP and others, and usually the message indicates that one of the communications protocols defined to DB2 Connect is not configured properly.

Often the cause might be that the DB2COMM profile variable is not defined, or is defined incorrectly. Generally, the problem is the result of a mismatch between the DB2COMM variable and names defined in the database manager configuration (for example, svcename or nname).

One possible scenario is having a previously successful connection, then getting the SQL5043 error message, while none of the configuration has changed. This could occur using the TCP/IP protocol, when the remote system abnormally terminates the connection for some reason. When this happens, a connection might still appear to exist on the client, and it might become possible to restore the connection without further intervention by issuing the commands shown below.

Most likely, one of the clients connecting to the DB2 Connect server still has a handle on the TCP/IP port. On each client machine that is connected to the DB2 Connect server, enter the following commands:

```
db2 terminate
db2stop
```

SQL30020

Symptom

SQL30020N Execution failed because of a Distributed Protocol Error that will affect the successful execution of subsequent commands and SQL statements.

Solutions

Service should be contacted with this error. Run the db2support command before contacting service.

SQL30060

Symptom

SQL30060N "*<authorization-ID>*" does not have the privilege to perform operation "*<operation>*".

Solution

When connecting to DB2 for OS/390® and z/OS, the Communications Database (CDB) tables have not been updated properly.

SQL30061

Symptom

Connecting to the wrong host or System i database server location - no target database can be found.

Solution

The wrong server database name might be specified in the DCS directory entry. When this occurs, SQLCODE -30061 is returned to the application.

Check the DB2 node, database, and DCS directory entries. The target database name field in the DCS directory entry must correspond to the name of the database based on the platform. For example, for a DB2 Universal Database™ for z/OS and OS/390 database, the name to be used should be the same as that used in the Boot Strap Data Set (BSDS) "LOCATION=*locname*" field, which is also provided in the DSNL004I message (LOCATION=*location*) when the Distributed Data Facility (DDF) is started.

The correct commands for a TCP/IP node are:

```
db2 catalog tcpip node <node_name> remote <host_name_or_address>
             server <port_no_or_service_name>
db2 catalog dcs database <local_name> as <real_db_name>
db2 catalog database <local_name> as <alias> at <node node_name>
             authentication server
```

To connect to the database you then issue:

```
db2 connect to <alias> user <user_name> using <password>
```

SQL30081N with Return Code 79

Symptom

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".
Location
where the error was detected: "".  Communication function
detecting the error:
"connect".  Protocol specific error code(s): "79", "*", "*".
SQLSTATE=08001
```

Solution(s)

This error can occur in the case of a remote client failing to connect to a DB2 Connect server. It can also occur when connecting from the DB2 Connect server to a host or System i database server.

1. The `DB2COMM` profile variable might be set incorrectly on the DB2 Connect server. Check this. For example, the command `db2set db2comm=tcpip` should appear in `sqllib/db2profile` when running DB2 Enterprise Server Edition on AIX.
2. There might be a mismatch between the TCP/IP service name and port number specifications at the IBM data server client and the DB2 Connect server. Verify the entries in the TCP/IP `services` files on both machines.
3. Check that DB2 is started on the DB2 Connect server. Set the Database Manager Configuration `diaglevel` to 4, using the command:

   ```
   db2 update dbm cfg using diaglevel 4
   ```

   After stopping and restarting DB2, look in the `db2diag.log` file to check that DB2 TCP/IP communications have been started. You should see output similar to the following:

   ```
   2001-02-03-12.41.04.861119   Instance:svtdbm2   Node:00
   PID:86496(db2sysc)   Appid:none
   common_communication  sqlcctcp_start_listen   Probe:80
   DIA3000I "TCPIP" protocol support was successfully started.
   ```

SQL30081N with Protocol Specific Error Code 10032

Symptom

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".
Location
where the error was detected: "9.21.85.159".  Communication
```

```
function detecting
the error: "send".  Protocol specific error code(s): "10032",
"*", "*".
SQLSTATE=08001
```

Solution

This error message might be received when trying to disconnect from a machine where TCP/IP communications have already failed. Correct the problem with the TCP/IP subsystem.

On most machines, simply restarting the TCP/IP protocol for the machine is the way to correct the problem. Occasionally, recycling the entire machine might be required.

SQL30082 RC=24 During CONNECT

Symptom

SQLCODE -30082 The username or the password supplied is incorrect.

Solution

Ensure that the correct password is provided on the CONNECT statement if necessary. Password not available to send to the target server database. A password has to be sent from the IBM data server client to the target server database. On certain platforms, for example AIX, the password can only be obtained if it is provided on the CONNECT statement.

Chapter 4. Tools for troubleshooting

Return codes internal to the database manager, tools that are part of the DB2 product, the different types of traces, and the tools that are part of the operating system are all used when troubleshooting your problems. Each tool provides data and information to assist you, and DB2 Support, while investigating a database problem.

Learning more about internal return codes

There are two types of internal return codes: ZRC values and ECF values. These are return codes that will generally only be visible in diagnostic tools intended for use by IBM Software Support. For example, they appear in DB2 trace output and in the db2diag.log file.

ZRC and ECF values basically serve the same purpose, but have slightly different formats. Each ZRC value has the following characteristics:

- Class name
- Component
- Reason code
- Associated SQLCODE
- SQLCA message tokens
- Description

However, ECF values consist of:

- Set name
- Product ID
- Component
- Description

ZRC and ECF values are typically negative numbers and are used to represent error conditions. ZRC values are grouped according to the type of error that they represent. These groupings are called ″classes″. For example, ZRC values that have names starting with “SQLZ_RC_MEMHEP” are generally errors related to insufficient memory. ECF values are similarly grouped into ″sets″.

An example of a db2diag.log entry containing a ZRC value is as follows:

```
2006-02-13-14.34.35.965000-300    I17502H435       LEVEL: Error
PID     : 940                TID  : 660    PROC : db2syscs.exe
INSTANCE: DB2                NODE : 000     DB   : SAMPLE
APPHDL  : 0-1433                APPID: *LOCAL.DB2.050120082811
FUNCTION: DB2 UDB, data protection, sqlpsize, probe:20
RETCODE : ZRC=0x860F000A=-2045837302=SQLO_FNEX "File not found."
          DIA8411C A file "" could not be found.
```

Full details about this ZRC value can be obtained using the db2diag command, for example:

```
c:\>db2diag -rc 0x860F000A

Input ZRC string '0x860F000A' parsed as 0x860F000A (-2045837302).

ZRC value to map: 0x860F000A (-2045837302)
        V7 Equivalent ZRC value: 0xFFFFE60A (-6646)
```

```
ZRC class :
        Critical Media Error (Class Index: 6)
Component:
        SQLO ; oper system services (Component Index: 15)
Reason Code:
        10 (0x000A)

Identifer:
        SQLO_FNEX
        SQLO_MOD_NOT_FOUND
Identifer (without component):
        SQLZ_RC_FNEX

Description:
        File not found.

Associated information:
        Sqlcode -980
SQL0980C  A disk error occurred.  Subsequent SQL statements cannot be
processed.

        Number of sqlca tokens : 0
        Diaglog message number: 8411
```

The same information is returned if you issue the commands db2diag -rc -2045837302 or db2diag -rc SQLO_FNEX.

An example of the output for an ECF return code is as follows:

```
c:\>db2diag -rc 0x90000076

Input ECF string '0x90000076' parsed as 0x90000076 (-1879048074).

ECF value to map: 0x90000076 (-1879048074)

ECF Set :
        setecf (Set index : 1)
Product :
        DB2 Common
Component:
        OSSe
Code:
        118 (0x0076)

Identifier:
        ECF_LIB_CANNOT_LOAD

Description:
        Cannot load the specified library
```

The most valuable troubleshooting information in the db2diag command output is the description and the associated information (for ZRC return codes only).

For a full listing of the ZRC or ECF values, use the commands db2diag -rc zrc and db2diag -rc ecf, respectively.

Overview of the db2dart tool

The db2dart command 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 issue the db2dart command 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, the db2dart utility will create a report file with the name databaseName.RPT. For single-partition database partition environments, the file is created in the current directory. For multiple-partition database partition environments, the file is created under a subdirectory in the diagnostic directory. The subdirectory is called DART####, where #### is the database partition number.

The db2dart utility accesses the data and metadata in a database by reading them directly from disk. Because of that, you should never run the tool against a database that still has active connections. If there are connections, the tool will not know about pages in the buffer pool or control structures in memory, for example, and might report false errors as a result. Similarly, if you run db2dart against a database that requires crash recovery or that has not completed rollforward recovery, similar inconsistencies might result due to the inconsistent nature of the data on disk.

Comparison of INSPECT and db2dart

The INSPECT command allows for the inspection of a database for architectural integrity, checking the pages of the database for page consistency. The INSPECT command checks that the structures of table objects and structures of table spaces are valid. Cross object validation conducts an online index to data consistency check. The db2dart command examines databases for architectural correctness and reports any encountered errors.

The INSPECT command is similar to the db2dart command in that it allows you to check databases, table spaces, and tables. A significant difference between the two commands is that the database needs to be deactivated before you run db2dart, whereas INSPECT requires a database connection and can be run while there are simultaneous active connections to the database.

If you do not deactivate the database, db2dart will yield unreliable results.

The following tables list the differences between the tests that are performed by the db2dart and INSPECT commands.

Table 1. Feature comparison of db2dart and INSPECT for table spaces

Tests performed	db2dart	INSPECT
SMS table spaces		
Check table space files	YES	NO
Validate contents of internal page header fields	YES	YES
DMS table spaces		
Check for extent maps pointed at by more than one object	YES	NO
Check every extent map page for consistency bit errors	NO	YES
Check every space map page for consistency bit errors	NO	YES

Table 1. Feature comparison of db2dart and INSPECT for table spaces (continued)

Tests performed	db2dart	INSPECT
Validate contents of internal page header fields	YES	YES
Verify that extent maps agree with table space maps	YES	NO

Table 2. Feature comparison of db2dart and INSPECT for data objects

Tests performed	db2dart	INSPECT
Check data objects for consistency bit errors	YES	YES
Check the contents of special control rows	YES	NO
Check the length and position of variable length columns	YES	NO
Check the LONG VARCHAR, LONG VARGRAPHIC, and large object (LOB) descriptors in table rows	YES	NO
Check the summary total pages, used pages and free space percentage	NO	YES
Validate contents of internal page header fields	YES	YES
Verify each row record type and its length	YES	YES
Verify that rows are not overlapping	YES	YES

Table 3. Feature comparison of db2dart and INSPECT for index objects

Tests performed	db2dart	INSPECT
Check for consistency bit errors	YES	YES
Check the location and length of the index key and whether there is overlapping	YES	YES
Check the ordering of keys in the index	YES	NO
Determine the summary total pages and used pages	NO	YES
Validate contents of internal page header fields	YES	YES
Verify the uniqueness of unique keys	YES	NO
Check for the existence of the data row for a given index entry	NO	YES

Table 3. Feature comparison of db2dart and INSPECT for index objects (continued)

Tests performed	db2dart	INSPECT
Verify each key to a data value	NO	YES

Table 4. Feature comparison of db2dart and INSPECT for block map objects

Tests performed	db2dart	INSPECT
Check for consistency bit errors	YES	YES
Determine the summary total pages and used pages	NO	YES
Validate contents of internal page header fields	YES	YES

Table 5. Feature comparison of db2dart and INSPECT for long field and LOB objects

Tests performed	db2dart	INSPECT
Check the allocation structures	YES	YES
Determine the summary total pages and used pages (for LOB objects only)	NO	YES

In addition, the following actions can be performed using the db2dart command:

- Format and dump data pages
- Format and dump index pages
- Format data rows to delimited ASCII
- Mark an index invalid

The INSPECT command cannot be used to perform those actions.

Analyzing db2diag.log files using db2diag

The primary log file intended for use by database and system administrators is the Administration Notification log. The db2diag.log file is intended for use by DB2 Support for troubleshooting purposes.

Administration notification log messages are also logged to the db2diag.log using a standardized message format.

The db2diag tool serves to filter and format the volume of information available in the db2diag.log. Filtering db2diag.log records can reduce the time required to locate the records needed when troubleshooting problems.

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

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

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

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

```
2006-02-15-19.31.36.114000-300 E21432H406         LEVEL: Error
PID     : 940                  TID  : 660         PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-1056               APPID: *LOCAL.DB2.060216003103
FUNCTION: DB2 UDB, base sys utilities, sqleDatabaseQuiesce, probe:2
MESSAGE : ADM7507W  Database quiesce request has completed successfully.
```

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

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

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

Note that this command could have been written a couple of different ways, including db2diag -l severe -pid 2200 -n 0,1,2,3. It should also be noted that the -g option specifies case-sensitive search, so here "Severe" will work but will fail if "severe" is used. These commands would successfully retrieve db2diag.log records which meet these requirements, such as:

```
2006-02-13-14.34.36.027000-300 I18366H421         LEVEL: Severe
PID     : 2200                 TID  : 660         PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-1433               APPID: *LOCAL.DB2.060213193043
FUNCTION: DB2 UDB, data management, sqldPoolCreate, probe:273
RETCODE : ZRC=0x8002003C=-2147352516=SQLB_BAD_CONTAINER_PATH
          "Bad container path"
```

Example 3: Formatting the db2diag tool output.

The following command filters all records occurring after January 1, 2006 containing non-severe and severe errors logged on partitions 0,1 or 2. It outputs the matched records such that the time stamp, partition number and level appear on the first line, pid, tid and instance name on the second line, and the error message follows thereafter:

```
db2diag -time 2006-01-01 -node "0,1,2" -level "Severe, Error" | db2diag -fmt
"Time: %{ts}
Partition: %node Message Level: %{level} \nPid: %{pid}  Tid: %{tid}
Instance: %{instance}\nMessage: @{msg}\n"
```

An example of the output produced is as follows:

```
Time: 2006-02-15-19.31.36.099000 Partition: 000 Message Level: Error
Pid: 940 Tid:940 Instance: DB2
Message: ADM7506W  Database quiesce has been requested.
```

For more information, issue the following commands:

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

Example 4: Filtering messages from different facilities.

The following examples show how to only see messages from a specific facility (or from all of them) from within the database manager. The supported facilities are:

- ALL which returns records from all facilities
- MAIN which returns records from DB2 general diagnostic logs such as the db2diag.log and the administration notification log
- OPSTATS which returns records related to optimizer statistics

To read messages from the MAIN facility:

```
db2diag -facility MAIN
```

To display messages from the OPSTATS facility and filter out records having a level of Severe:

```
db2diag -fac OPSTATS -level Severe
```

To display messages from all facilities available and filter out records having instance=harmistr and level=Error:

```
db2diag -fac all -g instance=harmistr,level=Error
```

To display all messages from the OPSTATS facility having a level of Error and then outputting the Timestamp and PID field in a specific format:

```
db2diag -fac opstats -level Error -fmt " Time :%{ts} Pid :%{ts}"
```

Displaying and altering the Global Registry (UNIX) using db2greg

The Global Registry exists only on UNIX and Linux platforms:

- For root installations, the Global Registry file is located at /var/db2/global.reg (/var/opt/db2/global.reg on HP-UX).
- For non-root installations, the Global Registry file is located at $HOME/sqllib/global.reg, where $HOME is the non-root user's home directory.

The Global Registry consists of three different record types:

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

You can view 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 can edit the Global Registry with the db2greg tool. Editing the Global Registry in root installations requires root authority.

You should only use the db2greg tool if requested to do so by DB2 Customer Support.

Identifying the version and service level of your product

The db2level command will help you determine the version and service level (build level and fix pack number) of your DB2 instance. To determine if your DB2 instance is at the latest service level, compare your db2level output to information in the fix pack download pages at the DB2 Support web site: http://www.ibm.com/software/data/db2/udb/support.html.

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

```
DB21085I  Instance "DB2" uses "32" bits and DB2 code release "SQL09010" with
level identifier "01010107".
Informational tokens are "DB2 v9.1.0.189", "n060119", "", and Fix Pack "0".
Product is installed at "c:\SQLLIB" with DB2 Copy Name "db2build".
```

The combination of the four informational tokens uniquely identify the precise service level of your DB2 instance. This information is essential when contacting IBM support for assistance.

For JDBC or SQLJ applications, if you are using the IBM DB2 Driver for SQLJ and JDBC, you can determine the level of the driver by running the db2jcc utility:

```
db2jcc -version

IBM DB2 JDBC Driver Architecture 2.3.63
```

Mimicking databases using db2look

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 might 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 might not contain the exact same set of data as the original database but you might 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 database partition groups, buffer pools, and table spaces will not be extracted. This is because they already exist in every database by default. If you want to mimic these, you must alter them yourself manually.

Bring up the file sample.ddl in a text editor. Since 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 might need to alter the path associated with the table space 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 the same options of 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: DB2INST1
-- Binding package automatically ...
-- Bind is successful
-- Binding package automatically ...
-- Bind is successful
```

The config.txt contains output similar to the following:

```
-- This CLP file was created using DB2LOOK Version 9.1
-- Timestamp: 2/16/2006 7:15:17 PM
-- Database Name: SAMPLE
-- Database Manager Version: DB2/NT Version 9.1.0
-- Database Codepage: 1252
-- Database Collating Sequence is: UNIQUE


CONNECT TO SAMPLE;

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

UPDATE DBM CFG USING cpuspeed 2.991513e-007;
UPDATE DBM CFG USING intra_parallel NO;
UPDATE DBM CFG USING comm_bandwidth 100.000000;
UPDATE DBM CFG USING federated NO;

...

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


COMMIT WORK;

CONNECT RESET;
```

Note: Only those parameters and variables that affect DB2 compiler will be included. If a registry variable that affects the compiler is set to its default value, it will not show up under ″Environment Variables settings″.

Listing DB2 products installed on your system (Linux and UNIX)

At least one DB2 Version 9 product must already be installed by a root user for a symbolic link to the db2ls command to be available in the /usr/local/bin directory.

With the ability to install multiple copies of DB2 products on your system and the flexibility to install DB2 products and features in the path of your choice, you need a tool to help you keep track of what is installed and where it is installed. On supported Linux and UNIX operating systems, the db2ls command lists the DB2 products and features installed on your system, including the DB2 Version 9 HTML documentation.

The db2ls command can be used to list:

- Where DB2 products are installed on your system and list the DB2 product level
- All or specific DB2 products and features in a particular installation path

The output that the db2ls command lists is different depending on the ID used:

- When the db2ls command is run with root authority, only root DB2 installations are queried.
- When the db2ls command is run with a non-root ID, root DB2 installations and the non-root installation owned by matching non-root ID are queried. DB2 installations owned by other non-root IDs are not queried.

The db2ls command is the only method to query a DB2 product. You *cannot* query DB2 products using Linux or UNIX operating system native utilities, such as pkginfo, rpm, SMIT, or swlist. Any existing scripts containing a native installation utility that you use to query and interface with DB2 installations will need to change.

You *cannot* use the db2ls command on Windows operating systems.

To list the path where DB2 products are installed on your system and list the DB2 product level, enter:

```
db2ls
```

The command lists the following information for each DB2 product installed on your system:

- Installation path
- Level
- Fix pack
- Special Install Number. This column is used by IBM DB2 Support.
- Installation date. This column shows when the DB2 product was last modified.
- Installer UID. This column shows the UID with which the DB2 product was installed.

To list information about DB2 products or features in a particular installation path the *q* parameter must be specified:

```
db2ls -q -p -b baseInstallDirectory
```

where:

- *q* specifies that you are querying a product or feature. This parameter is mandatory. If a DB2 Version 8 product is queried, a blank value is returned.
- *p* specifies that the listing displays products rather than listing the features.
- *b* specifies the installation directory of the product or feature. This parameter is mandatory if you are not running the command from the installation directory.

Depending on the parameters provided, the command lists the following information:

- Installation path. This is specified only once, not for each feature.
- The following information is displayed:
  - Response file ID for the installed feature, or if the *p* option is specified, the response file ID for the installed product. For example, ENTERPRISE_SERVER_EDITION.
  - Feature name, or if the *p* option is specified, product name.

  - Product version, release, modification level, fix pack level (VRMF). For example, 9.5.0.0
  - Fix pack, if applicable. For example, if Fix Pack 1 is installed, the value displayed is 1. This includes interim fix packs, such as Fix Pack 1a.
- If any of the product's VRMF information do not match, a warning message displays at the end of the output listing. The message suggests the fix pack to apply.

Monitoring and troubleshooting using db2pd

The db2pd tool is used for troubleshooting because it can return quick and immediate information from the DB2 memory sets.

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

If you want to capture information about the database management system when a specific SQLCODE, ZRC code or ECF code occurs, this can be accomplished using the db2pdcfg -catch command. When the errors are caught, the db2cos (callout script) is launched. The db2cos file can be dynamically altered to run any db2pd command, operating system command, or any other command needed to solve the problem. The template db2cos file is located in sqllib/bin on UNIX and Linux. On the Windows operating system, db2cos is located in the $DB2PATH\bin directory.

What follows is a collection of examples in which db2pd can be used to expedite troubleshooting.

Scenario 1: Diagnosing a lockwait.

Use the db2pd -db <database name> -locks -transactions -applications -dynamic command to get the following results:

```
Locks:
Address            TranHdl Lockname                   Type Mode Sts Owner Dur HldCnt Att    ReleaseFlg
0x07800000202E5238 3       00020002000000040000000052 Row  ..X  G   3     1   0      0x0000 0x40000000
0x07800000202E4668 2       00020002000000040000000052 Row  ..X  W*  2     1   0      0x0000 0x40000000
```

For the database that you specified using the -db database name option, the first results show the locks for that database. We can see that TranHdl 2 is waiting on a lock held by TranHdl 3.

```
Transactions:
Address            AppHandl [nod-index] TranHdl Locks State Tflag      Tflag2     Firstlsn       Lastlsn        LogSpace SpaceReserved TID            AxRegCnt GXID
0x0780000020251B80 11       [000-00011] 2       4     READ  0x00000000 0x00000000 0x000000000000 0x000000000000 0        0             0x0000000000B7 1        0
0x0780000020252900 12       [000-00012] 3       4     WRITE 0x00000000 0x00000000 0x000000FA000C 0x000000FA000C 113      154           0x0000000000B8 1        0
```

We can see that TranHdl 2 is associated with AppHandl 11 and TranHdl 3 is associated with AppHandl 12.

```
Applications:
Address            AppHandl [nod-index] NumAgents CoorPid Status         C-AnchID C-StmtUID L-AnchID L-StmtUID Appid

0x07800000006879E0 12       [000-00012] 1         1073336 UOW-Waiting    0        0         17       1         *LOCAL.burford.060303225602
0x0780000000685E80 11       [000-00011] 1         1040570 UOW-Executing  17       1         94       1         *LOCAL.burford.060303225601
```

We can see that AppHandl 12 last ran dynamic statement 17, 1. ApplHandl 11 is currently running dynamic statement 17, 1 and last ran statement 94, 1.

```
Dynamic SQL Statements:
Address            AnchID StmtUID NumEnv NumVar NumRef NumExe Text
0x07800000209FD800 17     1       1      1      2      2      update pdtest set c1 = 5
0x07800000209FCCC0 94     1       1      1      2      2      set lock mode to wait 1
```

We can see that the text column shows the SQL statements that are associated with the lock timeout.

Scenario 2: Using the -wlocks option to capture all the locks being waited on

In the sample output below, application 1 (AppHandl 47) is performing an insert, and application 2 (AppHandl 46) is selecting on that table.

```
venus@boson:/home/venus =>db2pd -wlocks -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:22

Locks being waited on :
AppHandl [nod-index] TranHdl    Lockname                   Type    Mode Conv Sts CoorEDU    AppName  AuthID   AppID
47       [000-00047] 8          00020004000000000840000652 Row     ..X       G   5160       db2bp    VENUS    *LOCAL.venus.071207213730
46       [000-00046] 2          00020004000000000840000652 Row     .NS       W   5913       db2bp    VENUS    *LOCAL.venus.071207213658
```

Scenario 3: Using the -apinfo option to capture detailed run time information about the lock owner and the lock waiter

The sample output below is captured under the same conditions as Scenario 2 above.

```
venus@boson:/home/venus =>db2pd -apinfo 47 -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:30

Application :
  Address :                0x0780000001676480
  AppHandl [nod-index] :   47       [000-00047]
  Application PID :        876558
  Application Node Name :  boson
  IP Address:              n/a
  Connection Start Time :  (1197063450)Fri Dec  7 16:37:30 2007
  Client User ID :         venus
  System Auth ID :         VENUS
  Coordinator EDU ID :     5160
  Coordinator Partition :  0
  Number of Agents :       1
  Locks timeout value :    4294967294 seconds
  Locks Escalation :       No
  Workload ID :            1
  Workload Occurrence ID : 2
  Trusted Context :        n/a
  Connection Trust Type :  non trusted
  Role Inherited :         n/a
  Application Status :     UOW-Waiting
  Application Name :       db2bp
  Application ID :         *LOCAL.venus.071207213730

  List of inactive statements of current UOW :
    UOW-ID :         2
    Activity ID :    1
    Package Schema : NULLID
    Package Name :   SQLC2G13
    Package Version :
    Section Number : 203
    SQL Type :       Dynamic
    Isolation :      CS
    Statement Type : DML, Insert/Update/Delete
    Statement :       insert into pdtest values 99


venus@boson:/home/venus =>db2pd -apinfo 46 -db pdtest

Database Partition 0 -- Database PDTEST -- Active -- Up 0 days 00:01:39

Application :
  Address :                0x0780000000D77A60
  AppHandl [nod-index] :   46       [000-00046]
  Application PID :        881102
  Application Node Name :  boson
  IP Address:              n/a
  Connection Start Time :  (1197063418)Fri Dec  7 16:36:58 2007
  Client User ID :         venus
  System Auth ID :         VENUS
  Coordinator EDU ID :     5913
  Coordinator Partition :  0
  Number of Agents :       1
  Locks timeout value :    4294967294 seconds
  Locks Escalation :       No
  Workload ID :            1
  Workload Occurrence ID : 1
  Trusted Context :        n/a
  Connection Trust Type :  non trusted
  Role Inherited :         n/a
  Application Status :     Lock-wait
  Application Name :       db2bp
  Application ID :         *LOCAL.venus.071207213658

  List of active statements :
   *UOW-ID :          3
    Activity ID :     1
    Package Schema :  NULLID
    Package Name :    SQLC2G13
    Package Version :
    Section Number :  201
    SQL Type :        Dynamic
    Isolation :       CS
    Statement Type :  DML, Select (blockable)
    Statement :       select * from pdtest
```

Scenario 4: Using the callout scripts when considering a locking problem

Find the db2cos output files. The location of the files is controlled by the database manager configuration parameter DIAGPATH. The contents of the output files will differ depending on what commands you enter in the db2cos file. An example of the output provided when the db2cos file contains a db2pd -db sample -locks command is as follows:

```
Lock Timeout Caught
Thu Feb 17 01:40:04 EST 2006
Instance DB2
Datbase: SAMPLE
Partition Number: 0
PID: 940
TID: 2136
Function: sqlplnfd
Component: lock manager
Probe: 999
Timestamp: 2006-02-17-01.40.04.106000
AppID: *LOCAL.DB2...
AppHdl:
...
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. A lock timeout can also occur when a lock is being converted to a higher mode. In those cases, you will not see a "W*" in the output, but rather a "C*". In this particular case, however, a lockwait has occurred. You can map the results to a transaction, application, agent, and even an SQL statement with the output provided by other db2pd commands in the db2cos file. You can narrow down the output or use other commands to collect the information you need. For example, you could change the db2pd command options to use the -locks wait option that only prints locks with a wait status. You could also put in -app, and -agent options if that is what you need.

Scenario 5: Mapping an application to a dynamic SQL statement

The command 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 statement.

```
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.burford.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 6: Monitoring memory usage.

The db2pd -memblock command can be useful when trying to understand memory usage. For example:

```
All memory blocks in DBMS set.

Address            PoolID   PoolName   BlockAge   Size(Bytes) I LOC   File
0x0780000000740068 62       resynch    2          112         1 1746  1583816485
0x0780000000725688 62       resynch    1          108864      1 127   1599127346
0x07800000001F4348 57       ostrack    6          5160048     1 3047  698130716
0x07800000001B5608 57       ostrack    5          240048      1 3034  698130716
0x07800000001A0068 57       ostrack    1          80          1 2970  698130716
0x07800000001A00E8 57       ostrack    2          240         1 2983  698130716
0x07800000001A0208 57       ostrack    3          80          1 2999  698130716
0x07800000001A0288 57       ostrack    4          80          1 3009  698130716
0x0780000000700068 70       apmh       1          360         1 1024  3878879032
0x07800000007001E8 70       apmh       2          48          1 914   1937674139
0x0780000000700248 70       apmh       3          32          1 1000  1937674139
...
```

This is followed by the sorted 'per-pool' output:

```
Memory blocks sorted by size for ostrack pool:
PoolID     PoolName   TotalSize(Bytes)     TotalCount LOC   File
57         ostrack    5160048              1          3047  698130716
57         ostrack    240048               1          3034  698130716
57         ostrack    240                  1          2983  698130716
57         ostrack    80                   1          2999  698130716
57         ostrack    80                   1          2970  698130716
57         ostrack    80                   1          3009  698130716
Total size for ostrack pool: 5400576 bytes
```

```
Memory blocks sorted by size for apmh pool:
PoolID     PoolName   TotalSize(Bytes)     TotalCount LOC   File
70         apmh       40200                2          121   2986298236
70         apmh       10016                1          308   1586829889
70         apmh       6096                 2          4014  1312473490
70         apmh       2516                 1          294   1586829889
70         apmh       496                  1          2192  1953793439
70         apmh       360                  1          1024  3878879032
70         apmh       176                  1          1608  1953793439
70         apmh       152                  1          2623  1583816485
70         apmh       48                   1          914   1937674139
70         apmh       32                   1          1000  1937674139
Total size for apmh pool: 60092 bytes
...
```

The final section of output sorts the consumers of memory for the entire set:

```
All memory consumers in DBMS memory set:
PoolID     PoolName   TotalSize(Bytes)     %Bytes TotalCount %Count LOC   File
57         ostrack    5160048              71.90  1          0.07   3047  698130716
50         sqlch      778496               10.85  1          0.07   202   2576467555
50         sqlch      271784               3.79   1          0.07   260   2576467555
57         ostrack    240048               3.34   1          0.07   3034  698130716
50         sqlch      144464               2.01   1          0.07   217   2576467555
62         resynch    108864               1.52   1          0.07   127   1599127346
72         eduah      108048               1.51   1          0.07   174   4210081592
69         krcbh      73640                1.03   5          0.36   547   4210081592
50         sqlch      43752                0.61   1          0.07   274   2576467555
70         apmh       40200                0.56   2          0.14   121   2986298236
69         krcbh      32992                0.46   1          0.07   838   698130716
50         sqlch      31000                0.43   31         2.20   633   3966224537
50         sqlch      25456                0.35   31         2.20   930   3966224537
52         kerh       15376                0.21   1          0.07   157   1193352763
50         sqlch      14697                0.20   1          0.07   345   2576467555
...
```

You can also report memory blocks for private memory on UNIX and Linux. For example:

```
db2pd -memb pid=159770

All memory blocks in Private set.

Address            PoolID     PoolName     BlockAge   Size(Bytes) I LOC   File
0x0000000110469068 88         private      1          2488        1 172   4283993058
0x0000000110469A48 88         private      2          1608        1 172   4283993058
0x000000011046A0A8 88         private      3          4928        1 172   4283993058
0x000000011046B408 88         private      4          7336        1 172   4283993058
0x000000011046D0C8 88         private      5          32          1 172   4283993058
0x000000011046D108 88         private      6          6728        1 172   4283993058
0x000000011046EB68 88         private      7          168         1 172   4283993058
0x000000011046EC28 88         private      8          24          1 172   4283993058
0x000000011046EC68 88         private      9          408         1 172   4283993058
0x000000011046EE28 88         private      10         1072        1 172   4283993058
0x000000011046F288 88         private      11         3464        1 172   4283993058
0x0000000110470028 88         private      12         80          1 172   4283993058
0x00000001104700A8 88         private      13         480         1 1534  862348285
0x00000001104702A8 88         private      14         480         1 1939  862348285
0x0000000110499FA8 88         private      80         65551       1 1779  4231792244
Total set size: 94847 bytes

Memory blocks sorted by size:
PoolID     PoolName   TotalSize(Bytes)     TotalCount LOC   File
88         private    65551                1          1779  4231792244
88         private    28336                12         172   4283993058
88         private    480                  1          1939  862348285
88         private    480                  1          1534  862348285
Total set size: 94847 bytes
```

Scenario 7: Determine which application is using up your table space

Using db2pd -tcbstats, the number of Inserts can be identified for a table. Here is sample information for a user-defined global temporary table called TEMP1:

```
TCB Table Information:
Address            TbspaceID TableID PartID MasterTbs MasterTab TableName  SchemaNm ObjClass DataSize  LfSize LobSize XMLSize
0x0780000020B62AB0 3         2       n/a    3         2         TEMP1      SESSION  Temp     966       0      0       0

TCB Table Stats:
Address            TableName   Scans   UDI   PgReorgs   NoChgUpdts   Reads   FscrUpdates Inserts   Updates   Deletes   OvFlReads OvFlCrtes
0x0780000020B62AB0 TEMP1       0       0     0          0            0       0           43968     0         0          0        0
```

You can then obtain the information for table space 3 via the db2pd -tablespaces command. Sample output is as follows:

```
Tablespace 3 Configuration:
Address            Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe  LastConsecPg Name
0x0780000020B1B5A0 DMS  UsrTmp  4096   32       Yes  32       1     1         On  1        0          31           TEMPSPACE2

Tablespace 3 Statistics:
Address            TotalPgs   UsablePgs  UsedPgs    PndFreePgs FreePgs    HWM        State      MinRecTime NQuiescers
0x0780000020B1B5A0 5000       4960       1088       0          3872       1088       0x00000000 0          0

Tablespace 3 Autoresize Statistics:
Address            AS  AR  InitSize    IncSize     IIP MaxSize     LastResize      LRF
0x0780000020B1B5A0 No  No  0           0           No  0           None            No

Containers:
Address            ContainNum Type    TotalPgs   UseablePgs StripeSet  Container
0x0780000020B1DCC0 0          File    5000       4960       0          /home/db2inst1/tempspace2a
```

You would notice the space filling up by referring to the FreePgs column. As the free pages value decreases, there is less space available. Notice also that the values for FreePgs plus UsedPgs will equal the value of UsablePgs.

Once this is known, you can identify the dynamic SQL statement that is using the table 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')
```

Finally, you can map this to db2pd -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.db2inst1.050202160426
```

The anchor ID (AnchID) value resulting from the request for dynamic SQL statements in the previous use of db2pd is used with the request for the associated applications. The applications result shows that the last anchor ID (L-AnchID) value is the same as the anchor ID (AnchID) value. The results from one run of db2pd is used in the next run of db2pd.

The output from db2pd -agent will show the number of rows read (Rowsread column) and rows written (Rowswrtn column) by the application. These values will give you an idea of what the application has completed and what the application still has to complete.

```
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      db2inst1 db2bp    22         9588       NotSet
```

The values for AppHandl and AgentPid from the db2pd -agent request can map back to the corresponding values for AppHandl and CoorPiid from the db2pd -app request.

The steps are slightly different if you suspect that an internal temporary table is filling up the table space. You would still use db2pd -tcbstats to identify tables with large numbers of inserts. Here is sample information for an implicit temporary table:

```
TCB Table Information:
Address            TbspaceID TableID PartID MasterTbs MasterTab TableName          SchemaNm ObjClass    DataSize  ...
0x0780000020CC0D30 1         2       n/a    1         2         TEMP (00001,00002) <30>     <JMC Temp   2470      ...
0x0780000020CC14B0 1         3       n/a    1         3         TEMP (00001,00003) <31>     <JMC Temp   2367      ...
0x0780000020CC21B0 1         4       n/a    1         4         TEMP (00001,00004) <30>     <JMC Temp   1872      ...

TCB Table Stats:
Address            TableName          Scans    UDI    PgReorgs    NoChgUpdts Reads    FscrUpdates Inserts ...
0x0780000020CC0D30 TEMP (00001,00002) 0        0      0           0          0        0           43219   ...
0x0780000020CC14B0 TEMP (00001,00003) 0        0      0           0          0        0           42485   ...
0x0780000020CC21B0 TEMP (00001,00004) 0        0      0           0          0        0           0       ...
```

In this example, there are a large number of inserts for tables with the naming convention "TEMP (TbspaceID, TableID)". These are implicit temporary tables. The values in the SchemaNm column have a naming convention of the value for AppHandl concatenated with the value for SchemaNm, which makes it possible to identify the application doing the work.

You can then map that information to the output from db2pd -tablespaces to see the used space for table space 1. Take note of the UsedPgs in relationship to the UsablePgs in the table space statistics.

```
Tablespace Configuration:
Address            Id    Type Content PageSz ExtentSz Auto Prefetch BufID BufIDDisk FSC NumCntrs MaxStripe LastConsecPg Name
0x07800000203FB5A0 1     SMS  SysTmp  4096   32       Yes  320      1     1         On  10       0         31           TEMPSPACE1

Tablespace Statistics:
Address            Id    TotalPgs   UsablePgs  UsedPgs    PndFreePgs FreePgs    HWM    State      MinRecTime NQuiescers
0x07800000203FB5A0 1     6516       6516       6516       0          0          0      0x00000000 0          0

Tablespace Autoresize Statistics:
Address            Id    AS  AR  InitSize    IncSize     IIP MaxSize     LastResize  LRF
0x07800000203FB5A0 1     No  No  0           0           No  0           None        No

Containers:
...
```

You can then identify the application handles 30 and 31 (since these were seen in the -tcbstats output), using the command db2pd -app:

```
Applications:
Address            AppHandl [nod-index] NumAgents  CoorPid    Status         C-AnchID C-StmtUID  L-AnchID   L-StmtUID  Appid
0x07800000006FB880 31       [000-00031] 1          4784182    UOW-Waiting    0        0          107        1          *LOCAL.db2inst1.051215214142
0x07800000006F9CE0 30       [000-00030] 1          8966270    UOW-Executing  107      1          107        1          *LOCAL.db2inst1.051215214013
```

Finally, map this to the Dynamic SQL using the db2pd -dyn command:

```
Dynamic SQL Statements:
Address            AnchID StmtUID    NumEnv     NumVar     NumRef     NumExe     Text
0x0780000020B296C0 107    1          1          1          43         43         select c1, c2 from test group by c1,c2
```

Scenario 8: Monitoring recovery.

The command db2pd -recovery shows several counters that you can use to verify that 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  TotalWork
0x0000000200667160 1        Forward     Wed Feb  2 16:41:14 2005 2268098 bytes  Unknown
0x0000000200667258 2        Backward    NotStarted               0 bytes        Unknown
```

Scenario 9: Determining the amount of resources a transaction is using

The command db2pd -transactions provides the number of locks, the first log sequence number (LSN), the last LSN, the log space used, and the space reserved. This can be useful for understanding the behavior of a 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.

The command db2pd -logs is useful for monitoring log usage for a database. By watching the Pages Written output, you can determine whether the log usage is progressing.

```
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 a value of ″Failure″ indicating that the most recent log archive failed. Alternatively, 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, the Next Log to Archive value will be lower than the Current Log Number value. 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: Viewing the sysplex list

Without the db2pd -sysplex command, the only way to report the sysplex list is via a DB2 trace.

```
Sysplex List:
Alias:         HOST
Location Name: HOST1
Count:         1

IP Address      Port       Priority   Connections Status     PRDID
 1.2.34.56      400        1          0           0
```

Scenario 12: Generating stack traces

The db2pd -stack all command for Windows operating systems (-stack for UNIX operating systems) can be used to produce stack traces for all processes in the current database partition. You might want to use this command iteratively when you suspect that a process or thread is looping or hanging.

You can obtain the current call stack for a particular engine dispatchable unit (EDU) by issuing the command db2pd -stack <eduid>. For example:

```
db2pd -stack 137

  Attempting to dump stack trace for eduid 137.
  See current DIAGPATH for trapfile.
```

If the call stacks for all of the DB2 processes are desired, use the command db2pd -stack all, for example (on Windows operating systems):

```
db2pd -stack all

  Attempting to dump all stack traces for instance.
  See current DIAGPATH for trapfiles.
```

If you are using a partitioned database environment with multiple physical nodes, you can obtain the information from all of the partitions by using the command db2_all ″; db2pd -stack all″. If the partitions are all logical partitions on the same machine, however, a faster method is to use db2pd -alldbp -stacks.

Scenario 13: Viewing memory statistics for a database partition

The db2pd -dbptnmem command shows how much memory the DB2 server is currently consuming and, at a high level, which areas of the server are using that memory.

Here is an example of the output from db2pd -dbptnmem on an AIX machine:

```
Database Partition Memory Controller Statistics

Controller Automatic: Y
Memory Limit:   122931408 KB
Current usage:     651008 KB
HWM usage:         651008 KB
Cached memory:     231296 KB
```

Here are the descriptions of these data fields and columns:

- Controller Automatic: Y if the **instance_memory** configuration parameter is set to AUTOMATIC. This means that database manager automatically determines the upper boundary on memory consumption.
- Memory Limit: The DB2 server's upper bound of memory that can be consumed. It is the value of the **instance_memory** configuration parameter.

- Current usage: The amount of memory the server is currently consuming.
- HWM usage: The high water mark (HWM) or peak memory usage that has been consumed since the activation of the database partition (when the db2start command was run).
- Cached memory: How much of the current usage is not currently being used, but is cached for performance reasons for future memory requests.

The continuation of the sample output from the db2pd -dbptnmem on AIX is shown below.

```
Individual Memory Consumers:
Name           Mem Used (KB)   HWM Used (KB)   Cached (KB)
===========================================================
APPL-DBONE       160000          160000        159616
DBMS-name         38528           38528          3776
FMP_RESOURCES     22528           22528             0
PRIVATE           13120           13120           740
FCM_RESOURCES     10048           10048             0
LCL-p606416         128             128             0
DB-DBONE         406656          406656         67200
```

All registered “consumers” of memory within the DB2 server are listed with the amount of the total memory they are consuming. The column descriptions are:

- Name: A brief, distinguishing name of a “consumer” of memory. Examples include:
  - APPL-<dbname> for application memory consumed for database <dbname>
  - DBMS-xxx for global database manager memory requirements
  - FMP_RESOURCES for memory required to communicate with db2fmps
  - PRIVATE for miscellaneous private memory requirements
  - FCM_RESOURCES for Fast Communication Manager resources
  - LCL-<pid> for memory segment used to communicate with local applications
  - DB-<dbname> for database memory consumed for database <dbname>
- Mem Used (KB): How much memory is currently allotted to that consumer.
- HWM Used (KB): High-Water Mark, or Peak, memory that the consumer has used.
- Cached (KB): Of the Mem Used (KB), the amount of memory that is not currently being used but is immediately available for future memory allocations.

Collecting environment information using db2support

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 the db2support utility avoids possible user errors, as you do not need to manually type commands such as GET DATABASE CONFIGURATION FOR <database name> or LIST TABLESPACES SHOW DETAIL. Also, you do not require instructions on which commands to run or files to collect, therefore it takes less time to collect the data.

- Execute the command db2support -h to display the complete list of command options.
- Collect data using the appropriate db2support command.

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 (for example, SQL1092N) might result when the utility runs commands such as QUERY CLIENT or LIST ACTIVE DATABASES.

If you're using the db2support utility to help convey information to IBM support, run the db2support command while the system is experiencing the problem. That way the tool will collect timely information, such as operating system performance details. If you are unable to run the utility at the time of the problem, you can still issue the db2support command after the problem has stopped since some first occurrence data capture (FODC) diagnostic files are produced automatically.

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 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 is possible to connect to the database.

The db2support utility collects the following information under all conditions:
- db2diag.log
- All trap files
- locklist files
- Dump files
- Various system related files
- Output from various system commands
- db2cli.ini

Depending on the circumstances, the db2support utility might also collect:
- Active log files
- Buffer pool and table space (SQLSPCS.1 and SQLSPCS.2) control files (with -d option)
- Contents of the db2dump directory
- Extended system information (with -s option)
- database configuration settings (with -d option)
- database manager configuration settings files
- Log File Header file (with -d option)
- Recovery History File (with -d option)

The HTML report db2support.html will always include the following information:
- Problem record (PMR) number (if -n was specified)
- Operating system and level (for example, AIX 5.1)
- DB2 release information
- An indication of whether it is a 32- or 64-bit environment
- DB2 install path information
- Contents of db2nodes.cfg

- Number of CPUs and disks and how much memory
- List of databases in the instance
- Registry information and environment, including PATH and LIBPATH
- Disk freespace for current filesystem and inodes for UNIX
- Java SDK level
- Database Manager Configuration
- Listing of the database recovery history file
- ls -lR output (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 the -s option is specified:

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

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

- Client connection state
- Database and Database Manager Configuration (Database Configuration requires the -d option)
- CLI config
- Memory pool info (size and consumed). Complete data is collected if the -d option is used
- The result of the LIST ACTIVE DATABASES command
- The result of the LIST DCS APPLICATIONS command

The db2support.html file contains the following information if the -c has been specified and a connection to the database is successfully established:

- Number of user tables
- Approximate size of database 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 DATABASE PARTITION GROUPS command
- The result of the LIST DBPARTITIONNUMS 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

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.

Collecting a trace often has a detrimental effect on the performance 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 (for example, 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.
- 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 might 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 Traces

How to obtain a DB2 trace using an internal utility is discussed. Once a DB2 trace file is created from the trace data in the trace buffer, you will need to format the output to make it readable. The information within that file can be used by DB2 Support to address your particular problem.

Obtaining a DB2 trace using db2trc

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.

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

In general, DB2 Support and development teams use DB2 traces for troubleshooting. You might run a trace to gain 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 are asked to obtain them.

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

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
```

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.

When turning trace on, the most important option 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 is usually 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 might 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 HP-UX, Linux and Solaris platforms, if the trace is turned off after the instance has been started, a very small buffer will be used the next time the trace is started regardless of the size specified. For example, yesterday you turned trace on by using db2trc on -l 8m, then collected a trace, and then turned the trace off (db2trc off). Today you wish to run a trace with the memory buffer set for 32 megabytes (db2trc on -l 32m) without bringing the instance down and restarting. You will find that in this case trace will only get a small buffer. To effectively run a trace on these platforms, turn the trace on before starting the instance with the size buffer you need and “clear” the buffer as necessary afterwards.

Dumping a DB2 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 by the command db2trc dmp is in binary format and 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
```

where example.trc is a binary file that was produced using the dmp option.

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 might 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 name of 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 -i option is specified), 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.

There are options available when formatting a binary file into a readable text file. For example, you can use `db2trc fmt -xml trace.dmp trace.fmt` to convert the binary data and output the result into an xml parsable format. Additional options are shown in the detailed description of the trace command (db2trc).

Another thing to be aware of is that on Linux and UNIX operating systems, 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 database partition 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 the common sequence of db2trc commands:

```
   db2trc on -l 8M
   db2trc clr
   <Execute problem recreation commands>
   db2trc dump db2trc.dmp
   db2trc off
   db2trc flw db2trc.dmp <filename>.flw
   db2trc fmt db2trc.dmp <filename>.fmt
   db2trc fmt -c db2trc.dmp <filename>.fmtc
```

DRDA Traces

Distributed Relational Database Architecture™ (DRDA) defines how data is passed between databases linked by common structures. DB2 Connect uses DRDA to link database client workstations to databases on host or System i™ machines. If you experience problems with such an environment, you will need to know about DRDA trace files, the utility used to create the trace, the output generated, and how to analyze the output. You are also shown DRDA trace samples. Finally, there is a recommendation to check subsequent send and receive buffers because they may contain information that is relevant to the problem you are experiencing.

DRDA trace files

Before analyzing DRDA traces, you need to understand that DRDA is an open standard for the definition of data and communication structures. For example, 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.org.

The **db2drdat** utility records the data interchanged between a DRDA Application Requestor (AR) and a DB2 DRDA Application Server (AS) (for example between DB2 Connect and a host or Series i database server).

Trace utility

The db2drdat utility records the data interchanged between the DB2 Connect server (on behalf of the IBM data server client) and the host or System i database server.

As a database administrator (or application developer), you might find it useful to understand how this flow of data works, because this knowledge can help you determine the origin of a particular problem. Suppose you found yourself in the following situation: you issue a `CONNECT TO` database statement for a host or System i 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 System i database server management system, you might 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.

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

Trace output

The db2drdat utility writes the following information to *tracefile*:

- -r
  - Type of DRDA reply/object
  - Receive buffer
- -s
  - Type of DRDA request
  - Send buffer
- -c
  - SQLCA
- TCP/IP error information
  - Receive function return code
  - Severity
  - Protocol used
  - API used
  - Function
  - Error number.

Note:

1. A value of zero for the exit code indicates that the command completed successfully, and a non-zero value indicates that it did not.
2. The fields returned vary based on the API used.
3. The fields returned vary based on the platform on which DB2 Connect is running, even for the same API.
4. If the db2drdat command sends the output to a file that already exists, the old file will be erased unless the permissions on the file do not allow it to be erased.

Trace output file analysis

The following information is captured in a db2drdat trace :

- The process ID (PID) of the client application
- The RDB_NAME cataloged in the database connection services (DCS) directory
- The DB2 Connect CCSID(s)
- The host or System i database server CCSID(s)
- The host or System i database server management system with which the DB2 Connect system is communicating.

The first buffer contains the Exchange Server Attributes (EXCSAT) and Access RDB (ACCRDB) commands sent to the host or System i database server management system. It sends these commands as a result of a `CONNECT TO` database command. The next buffer contains the reply that DB2 Connect received from the host or

System i database server management system. It contains an Exchange Server Attributes Reply Data (EXCSATRD) and an Access RDB Reply Message (ACCRDBRM).

EXCSAT

The EXCSAT command contains the workstation name of the client specified by the Server Name (SRVNAM) object, which is code point X'116D', according to DDM specification. The EXCSAT command is found in the first buffer. Within the EXCSAT command, the values X'9481A292' (coded in CCSID 500) are translated to *mask* once the X'116D' is removed.

The EXCSAT command also contains the EXTNAM (External Name) object, which is often placed in diagnostic information on the host or System i database management system. It consists of a 20-byte application ID followed by an 8-byte process ID (or 4-byte process ID and 4-byte thread ID). It is represented by code point X'115E', and in this example its value is `db2bp` padded with blanks followed by `000C50CC`. On a Linux or UNIX IBM data server client, this value can be correlated with the ps command, which returns process status information about active processes to standard output.

ACCRDB

The ACCRDB command contains the RDB_NAME in the RDBNAM object, which is code point X'2110'. The ACCRDB command follows the EXCSAT command in the first buffer. Within the ACCRDB command, the values X'E2E3D3C5C3F1' are translated to STLEC1 once the X'2110' is removed. This corresponds to the target database name field in the DCS directory.

The accounting string has code point X'2104'.

The code set configured for the DB2 Connect workstation is shown by locating the CCSID object CCSIDSBC (CCSID for single-byte characters) with code point X'119C' in the ACCRDB command. In this example, the CCSIDSBC is X'0333', which is 819.

The additional objects CCSIDDBC (CCSID for double-byte characters) and CCSIDMBC (CCSID for mixed-byte characters), with code points X'119D' and X'119E' respectively, are also present in the ACCRDB command. In this example, the CCSIDDBC is X'04B0', which is 1200, and the CCSIDMBC is X'0333', which is 819, respectively.

EXCSATRD and ACCRDBRM

CCSID values are also returned from the host or System i database server in the Access RDB Reply Message (ACCRDBRM) within the second buffer. This buffer contains the EXCSATRD followed by the ACCRDBRM. The example output file contains two CCSID values for the host or System i database server system. The values are 1208 (for both single-byte and mixed byte characters) and 1200 (for double-byte characters).

If DB2 Connect does not recognize the code page coming back from the host or System i database server, SQLCODE -332 will be returned to the user with the source and target code pages. If the host or System i database server doesn't recognize the code set sent from DB2 Connect, it will return VALNSPRM (Parameter Value Not Supported, with DDM code point X'1252'), which gets translated into SQLCODE -332 for the user.

The ACCRDBRM also contains the parameter PRDID (Product-specific Identifier, with code point X'112E'). The value is X'C4E2D5F0F8F0F1F5' which is DSN08015 in EBCDIC. According to standards, DSN is DB2

Universal Database for z/OS and OS/390. The version number is also indicated. ARI is DB2 Server for VSE & VM, SQL is DB2 database or DB2 Connect, and QSQ is DB2 for i5/OS.

Trace output file samples

The following figures show sample output illustrating some DRDA data streams exchanged between DB2 Connect workstations and a host or System i database server. From the user's viewpoint, a `CONNECT TO` database command has been issued using the command line processor (CLP).

Figure 2 on page 83 uses DB2 Connect Enterprise Edition Version 9.1 and DB2 Universal Database (UDB) for z/OS Version 8 over a TCP/IP connection.

```
1 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 0 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 233
2 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 19532 probe 1177
 bytes 250

 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  00C3D041000100BD 1041007F115E8482   ...A.....A...^..  .C}........".;db
  0010  F282974040404040 4040404040404040   ...@@@@@@@@@@@@@  2bp
  0020  4040F0F0F0C3F5F0 C3C3F0F0F0000000   @@..............    000C50CC000...
  0030  0000000000000000 0000000000000000   ................  ................
  0040  0000000000000000 000000000060F0F0   .............`..  .............-00
  0050  F0F1A2A495404040 4040404040404040   .....@@@@@@@@@@@  01sun
  0060  4040404040404040 4040404040404040   @@@@@@@@@@@@@@@@
  0070  C4C5C3E5F8404040 F0A2A49540404040   .....@@@....@@@@  DECV8   0sun
  0080  4040404040404040 4000181404140300   @@@@@@@@@.......         .......
  0090  0724070008147400 05240F0008144000   .$....t..$....@.  .............. .
  00A0  08000E1147D8C4C2 F261C1C9E7F6F400   ....G....a......  .....QDB2/AIX64.
  00B0  08116D9481A29200 0C115AE2D8D3F0F9   ..m.......Z.....  .._mask...]SQL09
  00C0  F0F0F0                              ...               000

           ACCSEC RQSDSS                    (ASCII)           (EBCDIC)
        0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0026D00100020020 106D000611A20003   .&..... .m......  ..}......_...s..
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  404040404040                        @@@@@@
3 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110546200 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 105
4 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110549755 probe 1178
 bytes 122

 RECEIVE BUFFER(AR):

           EXCSATRD OBJDSS                  (ASCII)           (EBCDIC)
        0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0059D04300010053 1443000F115EE5F8   .Y.C...S.C...^..  ..}..........;V8
  0010  F1C14BE2E3D3C5C3 F100181404140300   ..K.............  1A.STLEC1.......
  0020  0724070007147400 05240F0007144000   .$....t..$....@.  .............. .
  0030  0700081147D8C4C2 F20014116DE2E3D3   ....G.......m...  .....QDB2..._STL
  0040  C5C3F14040404040 4040404040000C11   ...@@@@@@@@@@...  EC1          ...
  0050  5AC4E2D5F0F8F0F1 F5                 Z........         ]DSN08015

           ACCSECRD OBJDSS                  (ASCII)           (EBCDIC)
        0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0010D0030002000A 14AC000611A20003   ................  ..}..........s..
5 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110656806 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 233
```

Figure 2. Example of Trace Output (TCP/IP connection)

```
6 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110659711 probe 1177
 bytes 250

 SEND BUFFER(AR):

            SECCHK RQSDSS                     (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  003CD04100010036 106E000611A20003   .<.A...6.n......  ..}......>...s..
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  404040404040000C 11A1D9858799F485   @@@@@@..........        ....Regr4e
  0030  A599000A11A09585 A6A39695           ............      vr....newton

            ACCRDB RQSDSS                     (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  00ADD001000200A7 20010006210F2407   ........ ...!.$.  ..}....x........
  0010  00172135C7F9F1C1 F0C4F3C14BD7C1F8   ..!5........K...  ....G91A0D3A.PA8
  0020  F806030221064600 162110E2E3D3C5C3   ....!.F..!......  8..........STLEC
  0030  F140404040404040 4040404040000C11   .@@@@@@@@@@@@...  1             ...
  0040  2EE2D8D3F0F9F0F0 F0000D002FD8E3C4   ............/...  .SQL09000....QTD
  0050  E2D8D3C1E2C30016 00350006119C0333   .........5.....3  SQLASC..........
  0060  0006119D04B00006 119E0333003C2104   ...........3.
7 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259908001 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 176
8 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259911584 probe 1178
 bytes 193

 RECEIVE BUFFER(AR):

            SECCHKRM RPYDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0015D0420001000F 1219000611490000   ...B.........I..  ..}.............
  0010  000511A400                          .....             ...u.

            ACCRDBRM RPYDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  009BD00200020095 2201000611490000   ........"....I..  ..}....n........
  0010  000D002FD8E3C4E2 D8D3F3F7F0000C11   .../............  ....QTDSQL370...
  0020  2EC4E2D5F0F8F0F1 F500160035000611   ............5...  .DSN08015.......
  0030  9C04B80006119E04 B80006119D04B000   ................  ................
  0040  0C11A0D5C5E6E3D6 D540400006212524   .........@@..!%$  ...NEWTON  .....
  0050  34001E244E000624 4C00010014244D00   4..$N..$L....$M.  ....+...<.....(.
  0060  06244FFFFF000A11 E8091E768301BE00   .$O........v....  ..!.....Y...c...
  0070  2221030000000005 68B3B8C7F9F1C1F0   "!......h.......  ...........G91A0
  0080  C4F3C1D7C1F8F840 4040400603022106   .......@@@@...!.  D3APA88    .....
  0090  46000A11E8091E76 831389             F......v...       ....Y...c.i
9 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364420503 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 10
```

Figure 3. Example of Trace Output (TCP/IP connection) continued

```
10 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364440751 probe 1177
 bytes 27

 SEND BUFFER(AR):

            RDBCMM RQSDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200E               ........ .        ..}.......
11 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475009631 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54
12 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475014579 probe 1178
 bytes 71

 RECEIVE BUFFER(AR):

            ENDUOWRM RPYDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  002BD05200010025 220C000611490004   .+.R...%"....I..  ..}.............
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  4040404040400005 211501             @@@@@@..!..             .....

            SQLCARD OBJDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000BD00300010005 2408FF             ........$..       ..}........
13 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721710319 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 126
14 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721727276 probe 1177
 bytes 143

 SEND BUFFER(AR):

            EXCSQLIMM RQSDSS                 (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0053D0510001004D 200A00442113E2E3   .S.Q...M ..D!...  ..}....(......ST
  0010  D3C5C3F140404040 4040404040404040   ....@@@@@@@@@@@@  LEC1
  0020  D5E4D3D3C9C44040 4040404040404040   ......@@@@@@@@@@  NULLID
  0030  4040E2D8D3C3F2C6 F0C1404040404040   @@........@@@@@@    SQLC2F0A
  0040  4040404041414141 41484C5600CB0005   @@@@AAAAAHLV....      ......<.....
  0050  2105F1                              !..               ..1

            SQLSTT OBJDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  002BD00300010025 2414000000001B64   .+.....%$......d  ..}.............
  0010  656C657465206672 6F6D206464637375   elete from ddcsu  .%......?_......
  0020  73312E6D79746162 6C65FF             s1.mytable.       ..._`./.%..
15 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832901261 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 102
```

Figure 4. Example of Trace Output (TCP/IP connection) continued

```
16 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832906528 probe 1178
 bytes 119

 RECEIVE BUFFER(AR):

           SQLCARD OBJDSS                   (ASCII)           (EBCDIC)
        0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0066D00300010060 240800FFFFFF3434   .f.....`$.....44  ..}....-........
  0010  3237303444534E58 4F544C2000FFFFFE   2704DSNXOTL ....  ......+.!.<.....
  0020  0C00000000000000 00FFFFFFFF000000   ................  ................
  0030  0000000000572020 2057202020202020   .....W   W        ................
  0040  001053544C454331 2020202020202020   ..STLEC1          ....<...........
  0050  2020000F44444353 5553312E4D595441     ..DDCSUS1.MYTA  ............(...
  0060  424C450000FF                        BLE...            .<....
17 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833156953 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 10
18 data DB2 UDB DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833159843 probe 1177
 bytes 27

 SEND BUFFER(AR):

           RDBRLLBCK RQSDSS                 (ASCII)           (EBCDIC)
        0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200F               ........ .        ..}.......
19 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943302832 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54
20 data DB2 UDB DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943306288 probe 1178
 bytes 71

 RECEIVE BUFFER(AR):

           ENDUOWRM RPYDSS                  (ASCII)           (EBCDIC)
        0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  002BD05200010025 220C000611490004   .+.R...%"....I..  ..}.............
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  4040404040400005 211502             @@@@@@..!..             .....

           SQLCARD OBJDSS                   (ASCII)           (EBCDIC)
        0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000BD00300010005 2408FF             ........$..       ..}........
```

Figure 5. Example of Trace Output (TCP/IP connection) continued

Subsequent buffer information for DRDA traces

You can analyze subsequent send and receive buffers for additional information. The next request contains a commit. The commit command instructs the host or System i database server management system to commit the current unit of work. The fourth buffer is received from the host or System i database server database management system as a result of a commit or rollback. It contains the End Unit of Work Reply Message (ENDUOWRM), which indicates that the current unit of work has ended.

In this example, trace entry 12 contains a null SQLCA, indicated by DDM code point X'2408' followed by X'FF'. A null SQLCA (X'2408FF') indicates success (SQLCODE 0).

Figure 2 on page 83 shows an example of a receive buffer containing an error SQLCA at trace entry 16.

Control Center traces

Before attempting to trace a problem in the Control Center, it is advisable to first ensure that the same problem does not occur when the equivalent actions are performed via explicit commands from the DB2 command prompt. Often when you are performing a task within the Control Center (or one of the other GUI tools which can be launched from the Control Center), you will see a "Show Command" button, which provides the exact syntax for the command which the tool will use. If that exact command succeeds from the DB2 command prompt, but fails when executed within the GUI tool, then it is appropriate to obtain a Control Center trace.

In order to obtain a trace of a problem which is only reproducible within the Control Center, launch the Control Center as follows:

```
db2cc -tf filename
```

This turns on the Control Center Trace and saves the output of the trace to the specified file. The output file is saved to `<DB2 install path>\sqllib\tools` on Windows and to `/home/<userid>/sqllib/tools` on UNIX and Linux.

Note: When the Control Center has been launched with tracing enabled, recreate the problem using as few steps as possible. Try to avoid clicking on unnecessary or unrelated items in the tool. Once you have recreated the problem, close the Control Center (and any other GUI tools which you opened to recreate the problem).

The resulting trace file will need to be sent to DB2 Support for analysis.

JDBC traces

Depending on the type of the JDBC driver you are using, there are different ways to obtain trace files for the applications or stored procedures you are running. These different ways are presented here.

Obtaining traces of applications that use the DB2 JDBC Type 2 Driver for Linux, UNIX and Windows

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

- a JDBC application which uses the DB2 JDBC Type 2 Driver for Linux, UNIX and Windows (DB2 JDBC Type 2 Driver)
- DB2 JDBC stored procedures.

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 Support. If you have keywords in your db2cli.ini file that are not documented, it is likely that they were recommended by DB2 Support. Internally, the DB2 JDBC Type 2 Driver makes use of the DB2 CLI driver for database access. For example, the Java getConnection() method is internally mapped by the DB2 JDBC Type 2 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.

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 Linux and UNIX:

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

2. Update the CLI configuration keywords. There are two methods to accomplish this:

   - Manually edit the db2cli.ini file.

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

        By default, the location of the DB2 CLI/ODBC configuration keyword file (db2cli.ini) is in the sqllib directory on Windows 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 might 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.

     b. Add the following section to the file (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 Linux or UNIX operating systems.

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

   - Use 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 Linux or UNIX operating systems.

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

   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.

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

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

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.

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.
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
   ```
7. Restart any applications that are running and tracing.
8. 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.

Obtaining traces of applications that use the DB2 Universal JDBC Driver

If you have an SQLJ or JDBC application that is using the DB2 Universal JDBC Driver, a JDBC trace can be enabled in several different ways:

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

CLI trace files

The CLI trace captures information about applications that access the DB2 CLI driver.

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 CLI application
- an ODBC application (since ODBC applications use the DB2 CLI interface to access DB2)
- DB2 CLI stored procedures
- JDBC applications and stored procedures

When diagnosing ODBC applications it is often easiest to determine the problem by using an ODBC trace or DB2 CLI trace. If you are using an ODBC driver manager, it will likely provide the capability to take an ODBC trace. Consult your driver manager documentation to determine how to enable ODBC tracing. DB2 CLI traces are DB2-specific and will often contain more information than a generic ODBC trace. Both traces are usually quite similar, listing entry and exit points for all CLI calls from an application; including any parameters and return codes to those calls.

The DB2 JDBC Type 2 Driver for Linux, UNIX and Windows (DB2 JDBC Type 2 Driver) depends on the DB2 CLI driver to access the database. Consequently, Java developers might also want to enable DB2 CLI tracing for additional information on how their applications interact with the database through the various software layers. DB2 JDBC and DB2 CLI trace options (though both set in the db2cli.ini file) are independent of each other.

Enabling CLI traces

A CLI 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 Support. If you have keywords in your db2cli.ini file that are not documented, it is likely that they were recommended by the DB2 support team.

By default, the location of the DB2 CLI/ODBC configuration keyword file is in the sqllib directory on Windows operating systems, and in the sqllib/cfg directory of the database instance running the CLI/ODBC applications on Linux and UNIX operating systems. If the ODBC Driver Manager is used to configure a User Data Source on Windows, a db2cli.ini can 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.

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 Linux and UNIX:

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

2. Update the CLI configuration keywords.

   This can be done by either manually editing the db2cli.ini file or 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]
      Trace=1
      TracePathName=<path>
      TraceComm=1
      TraceFlush=1
      TraceTimeStamp=1
      TraceScript=1
      ```

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

   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 Trace 1
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TracePathName <path>
   ```

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

   ```
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceComm 1
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceFlush 1
   db2 UPDATE CLI CFG FOR SECTION COMMON USING TraceTimeStamp 3
   ```

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
   ```

4. Restart the application.

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

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

5. Capture the error.

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

6. Disable the CLI trace.

   Set the **Trace** keyword to a value of zero in the [COMMON] section of the db2cli.ini manually, or issue:

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

   Then restart any applications that might be running and tracing.

7. Collect the trace information.

   The CLI trace files will be written to the path specified in the **TracePathName** keyword. The filenames generated have a format of p<pid>t<tid>.cli where <pid> is the operating system assigned process ID, and <tid> is a numerical counter (starting at 0) for each thread generated by the application process. For example, p1234t1.cli. If you are working with DB2 Support to diagnose a problem, they will want to see all of the files that were generated in the trace path.

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.

Interpreting input and output parameters in CLI trace files

As is the case with any regular function, DB2 CLI functions have input and output parameters. In a DB2 CLI trace, these input and output parameters can be seen, providing details about how each application is invoking a particular CLI API. The input and output parameters for any CLI function as shown in the CLI trace can be compared to the definition of that CLI function in the CLI Reference sections of the documentation.

Here is a snippet from a CLI trace file:

```
SQLConnect( hDbc=0:1, szDSN="sample", cbDSN=-3, szUID="",
            cbUID=-3, szAuthStr="", cbAuthStr=-3 )
    ---> Time elapsed - +6.960000E-004 seconds

SQLRETURN   SQLConnect       (
            SQLHDBC            ConnectionHandle,  /* hdbc */
            SQLCHAR       *FAR ServerName,        /* szDSN */
            SQLSMALLINT        NameLength1,       /* cbDSN */
            SQLCHAR       *FAR UserName,          /* szUID */
            SQLSMALLINT        NameLength2,       /* cbUID */
            SQLCHAR       *FAR Authentication,    /* szAuthStr */
            SQLSMALLINT        NameLength3);      /* cbAuthStr */
```

The initial call to the CLI function shows the input parameters and the values being assigned to them (as appropriate).

When CLI functions return, they show the resultant output parameters, for example:

```
SQLAllocStmt( phStmt=1:1 )
    <--- SQL_SUCCESS   Time elapsed - +4.444000E-003 seconds
```

In this case, the CLI function SQLAllocStmt() is returning an output parameter phStmt with a value of ″1:1″ (connection handle 1, statement handle 1).

Analyzing Dynamic SQL in CLI traces

DB2 CLI Traces also show how dynamic SQL is performed, via the declaration and use of parameter markers in SQLPrepare() and SQLBindParameter(). This gives you the ability to determine at runtime what SQL statements will be performed.

The following trace entry shows the preparation of the SQL statement (’?’ denotes a parameter marker):

```
 SQLPrepare( hStmt=1:1, pszSqlStr=
   "select * from employee where empno = ?",
   cbSqlStr=-3 )
    ---> Time elapsed - +1.648000E-003 seconds
( StmtOut="select * from employee where empno = ?" )
SQLPrepare( )
  <--- SQL_SUCCESS   Time elapsed - +5.929000E-003 seconds
```

The following trace entry shows the binding of the parameter marker as a CHAR with a maximum length of 7:

```
SQLBindParameter( hStmt=1:1, iPar=1, fParamType=SQL_PARAM_INPUT,
fCType=SQL_C_CHAR, fSQLType=SQL_CHAR, cbColDef=7, ibScale=0,
 rgbValue=&00854f28, cbValueMax=7, pcbValue=&00858534 )
    ---> Time elapsed - +1.348000E-003 seconds
SQLBindParameter( )
    <--- SQL_SUCCESS   Time elapsed - +7.607000E-003 seconds
```

The dynamic SQL statement is now executed. The rbgValue=″000010″ shows the value that was substituted for the parameter marker by the application at run time:

```
SQLExecute( hStmt=1:1 )
    ---> Time elapsed - +1.317000E-003 seconds
( iPar=1, fCType=SQL_C_CHAR, rgbValue="000010" - X"303030303130",
 pcbValue=6, piIndicatorPtr=6 )
    sqlccsend( ulBytes - 384 )
    sqlccsend( Handle - 14437216 )
    sqlccsend( ) - rc - 0, time elapsed - +1.915000E-003
    sqlccrecv( )
    sqlccrecv( ulBytes - 1053 ) - rc - 0, time elapsed - +8.808000E-003
SQLExecute( )
    <--- SQL_SUCCESS   Time elapsed - +2.213300E-002 seconds
```

Interpreting timing information in CLI traces

There are a few ways to gather timing information from a DB2 CLI trace. By default a CLI trace captures the time spent in the application since the last CLI API call was made on this particular thread. As well as the time spent in DB2, it includes the network time between the client and server. For example:

```
SQLAllocStmt( hDbc=0:1, phStmt=&0012ee48 )
    ---> Time elapsed - +3.964187E+000 seconds
```

(This time value indicates the time spent in the application since last CLI API was called)

```
SQLAllocStmt( phStmt=1:1 )
    <--- SQL_SUCCESS   Time elapsed - +4.444000E-003 seconds
```

(Since the function has completed, this time value indicates the time spent in DB2, including the network time)

The other way to capture timing information is using the CLI keyword: TraceTimeStamp. This keyword will generate a timestamp for every invocation and result of a DB2 CLI API call. The keyword has 4 display options: no timestamp information, processor ticks and ISO timestamp, processor ticks, or ISO timestamp.

This can be very useful when working with timing related problems such as CLI0125E - function sequence errors. It can also be helpful when attempting to determine which event happened first when working with multithreaded applications.

Interpreting unknown values in CLI traces

It is possible that a DB2 CLI function might return ″Unknown value″ as a value for an input parameter in a CLI trace. This can occur if the DB2 CLI driver is looking for something specific for that input parameter, yet the application provides a different value. For example, this can occur if you're following out-dated definitions of CLI functions or are using CLI functions which have been deprecated.

It is also possible that you could see a CLI function call return an ″Option value changed″ or a ″Keyset Parser Return Code″. This is a result of the keyset cursor displaying a message, such as when the cursor is being downgraded to a static cursor for some specific reason.

```
SQLExecDirect( hStmt=1:1, pszSqlStr="select * from org", cbSqlStr=-3 )
    ---> Time elapsed - +5.000000E-002 seconds
( StmtOut="select * from org" )
( COMMIT=0 )
( StmtOut=" SELECT A.TABSCHEMA, ...... )
( StmtOut=" SELECT A.TABSCHEMA, ...... )
( Keyset Parser Return Code=1100 )

SQLExecDirect( )
    <--- SQL_SUCCESS_WITH_INFO   Time elapsed - +1.06E+001 seconds
```

In the above CLI trace, the keyset parser has indicated a return code of 1100, which indicates that there is not a unique index or primary key for the table, and therefore a keyset cursor could not be created. These return codes are not externalized and thus at this point you would need to contact DB2 Support if you wanted further information about the meaning of the return code.

Calling SQLError or SQLDiagRec will indicate that the cursor type was changed. The application should then query the cursor type and concurrency to determine which attribute was changed.

Interpreting multi-threaded CLI trace output

CLI traces can trace multi-threaded applications. The best way to trace a multithreaded application is by using the CLI keyword: TracePathName. This will produce trace files named p<pid>t<tid>.cli where <tid> is the actual thread id of the application.

If you need to know what the actual thread id is, this information can be seen in the CLI Trace Header:

```
[ Process: 3500, Thread: 728 ]
[ Date & Time:          02/17/2006 04:28:02.238015 ]
[ Product:              QDB2/NT DB2 v9.1.0.190 ]
...
```

You can also trace a multithreaded application to one file, using the CLI keyword: TraceFileName. This method will generate one file of your choice, but can be cumbersome to read, as certain API's in one thread can be executed at the same time as another API in another thread, which could potentially cause some confusion when reviewing the trace.

It is usually recommended to turn TraceTimeStamp on so that you can determine the true sequence of events by looking at the time that a certain API was executed. This can be very useful for investigating problems where one thread caused a problem in another thread (for example, CLI0125E - Function sequence error).

Platform-specific tools

There are troubleshooting commands, performance monitoring utilities, and other methods of gathering diagnostic information that are associated with the platforms you are using. These tools are presented as part of your Windows operating system, or your Linux and UNIX operating systems.

Diagnostic tools (Windows)

The following diagnostic tools are available for Windows operating systems:

Event viewer, 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 might help in diagnosing a problem, and saves this information to a file. You must start this utility by typing `drwatson` on the command line.

Diagnostic tools (Linux and UNIX)

This section describes some essential commands for troubleshooting and performance monitoring on Linux and UNIX platforms. 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 system. Once the data is collected, it can be examined by someone who is familiar with the problem, or provided to DB2 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 database partition:

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

xmperf
For AIX systems using Motif, this command starts a graphical monitor that collects and displays system-related performance data. The monitor displays three-dimensional diagrams for each database partition 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.

spmon
If you are using system partitioning as part of the Parallel System Support Program (PSSP), you might need to check if the SP Switch is running on all workstations. To view the status of all database partitions, 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 database partition workstation to see if the switch is down. If the switch is down, there is an asterisk (*) beside the database partition name. For example:

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

The asterisk does not appear if the switch is up.

Troubleshooting commands (Linux and UNIX)

The following system commands are for all Linux and UNIX 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 This command is useful for tracing system calls in one or more processes.

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 system.

vmstat
This command is useful for determining if something is suspended or just taking a long time. You can monitor the paging rate, 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).

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).

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

system file
Available for Solaris operating system, 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 system machine.

Chapter 5. How to search effectively for known problems

There are many resources available that describe known problems, including DB2 APARs, whitepapers, IBM Redbooks™ publications, 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, you should have a clear understanding of your problem situation.

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. For example, 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 quotation marks) 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. For example, search for ″EFBIG″ instead of the error number 27.

In general, search terms that are successful often involve:

- Words that describe the command run
- Words that describe the symptoms
- Tokens from the diagnostics

Troubleshooting resources

A wide variety of troubleshooting information is available to assist you in using DB2 database 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), fix packs 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 6. Getting DB2 product fixes

Fix packs contain code updates and fixes for problems found by IBM during product testing and by customers as they use the product. How to find the latest fix pack and how to apply the fixes to your database environment are discussed.

Applying fix packs

It is recommended that you keep your DB2 environment running at the latest fix pack level to ensure problem-free operation. To install a fix pack successfully, perform all of the necessary pre-installation and post-installation tasks.

A DB2 fix pack contains updates and fixes for problems (Authorized Program Analysis Reports, or ″APARs″) found during testing at IBM, as well as fixes for problems reported by customers. Each fix pack contains an APARLIST.TXT file, which describes the fixes it contains.

Fix packs are cumulative. This means that the latest fix pack for any given version of DB2 contains all of the updates from previous fix packs for the same version of DB2.

The fix pack images available are:

- A single server image.

  The single server image contains the new and updated code required for all DB2 server products and the IBM Data Server Client. If more than one DB2 server product is installed in a single location, the DB2 server fix pack applies maintenance code updates to all the installed DB2 server products. The Data Server Client fix pack is contained within the one DB2 server fix pack (namely the fix pack that can service any one of the following server products: Enterprise Server Edition, Workgroup Server Edition, Express Edition, Personal Edition, Connect Enterprise Edition, Connect Application Server Edition, Connect Unlimited Edition for zSeries®, and Connect Unlimited Edition for i5/OS®). You can use the DB2 server fix pack to upgrade a Data Server Client.

  A single server image can also be used to install any of the DB2 database server products, at a particular fix pack level, with a DB2 try and buy license by default.
- A fix pack for each of the other DB2 database products.

  Use this fix pack only if you only have non-server database products or add-on products installed. For example, IBM Data Server Runtime Client or Query Patroller.

  Do not use this type of fix pack if the installed DB2 products are only DB2 server products or a Data Server Client. Instead, use the single server image fix pack.

  For Windows platforms, if you have more than one DB2 database product (which includes at least one product that is not a Data Server Client or a DB2 server) installed in a single DB2 copy, you must download and uncompress all of the corresponding product-specific fix packs before starting the fix pack installation process.
- A universal fix pack (on Linux or UNIX platforms only).

The universal fix pack services installations where more than one DB2 database product has been installed.

The universal fix pack is not needed if the installed DB2 products are only DB2 server products or a Data Server Client. In this case, the single server image fix pack should used.

Restrictions

- A DB2 Version 9.5 fix pack can only be applied to DB2 Version 9.5 general availability (GA) or fix pack level copies.
- All DB2 instances, DAS, and applications related to the DB2 copy being updated must be stopped before installing a fix pack.
- If you are using the database partitioning feature (DPF), prior to installing the fix pack, you must stop the database manager on all nodes. You must install the fix pack on the instance owning node and all other partitioned nodes. All computers participating in the instance must be upgraded to the same fix pack level.
- On Linux or UNIX operating systems:
  - If you have DB2 products on a Network File System (NFS), you must ensure the following are stopped completely before installing the fix pack: all instances, the DB2 administration server (DAS), interprocess communications (IPC), and applications on other machines using the same NFS mounted installation.
  - If the system commands fuser or lsof are not available, the installFixPack command cannot detect loaded DB2 files. You must ensure no DB2 files are loaded and provide an override option to install the fix pack. On UNIX, the fuser command is required to check for loaded files. On Linux, either the fuser command or lsof command is required.

    For details on the override option, see the installFixPack command.
- On client applications, after a fix pack has been applied, to perform autobind of applications, the user must have bind authority.
- Installation of a DB2 fix pack will not service IBM Data Studio Administration Console or IBM Data Studio.

For non-root installations on Linux or UNIX, root-based features (such as High Availability and operating system-based authentication) can be enabled using the db2rfe command. If root-based features were enabled after installing your DB2 product, you must rerun the db2rfe command each time a fix pack is applied in order to re-enable those features. For details, see the non-root related links below.

On Linux or UNIX operating systems, if national languages have been installed, you also require a separate national language fix pack. The national language fix pack can not be installed alone. A universal or product-specific fix pack must be applied at the same time and they must both be at the same fix pack level. For example, if you are applying a universal fix pack to non-English DB2 database products on Linux or UNIX, you must apply both the universal fix pack and the national language fix pack to update the DB2 database products.

If you have multiple DB2 copies on the same system, those copies can be at different version and fix pack levels. If you want to apply a fix pack to one or more DB2 copies, you must install the fix pack on those DB2 copies one by one.

Fix packs, test fixes and APARs

An Authorized Program Analysis Report (APAR) is a formal report of a problem caused by a suspected defect in a current unaltered release of an IBM program. APARs describe problems found during testing by IBM, as well as problems reported by customers.

The modified DB2 code that resolves the problem described in the APAR can be delivered in a fix pack or a test fix.

Fix pack
A fix pack is a cumulative collection of APAR fixes. In particular, fix packs address the APARs that arise between new releases of DB2. They are intended to allow you to move up to a specific maintenance level. Fix packs have the following characteristics:

- They are cumulative. Fix packs for a particular release of DB2 supersede or contain all of the APAR fixes shipped in previous fix packs for that release.
- They contain many APARs.
- They are published on the DB2 Technical Support Web site and are generally available to customers who have purchased products under the Passport Advantage® program.
- They are fully tested by IBM.
- They contain a Readme and a set of Release Notes.
  - The fix pack Readme provides instructions for installing and removing the fix pack.
  - The Release Notes contain information about changes to the product

Note: The status of an APAR is changed from ″Open″ to ″Closed as program error″ when the APAR fix is provided in a fix pack. You can determine the status of individual APARs by examining the APAR descriptions on the DB2 Technical Support Web site

Test fix
A test fix is a temporary solution that is supplied to specific customers for testing in response to a reported problem. Test fixes have the following characteristics:

- They usually contain a single APAR.
- They are obtained from DB2 Support and are not generally available to the public.
- They undergo limited IBM testing.
- They include minimal documentation, including a description of how the test fix should be applied, the APAR(s) it fixes, and instructions for the removal of the test fix.

Test fixes are supplied in situations where a new problem has been uncovered, there is no workaround or bypass for the problem and you cannot wait until the next fix pack becomes available. For example, if the problem is causing critical impact on your business, a test fix might be provided to alleviate the situation until the APAR is addressed in a fix pack.

It is recommended that you keep your DB2 environment running at the latest fix pack level to ensure problem-free operation. To receive notification of the availability of new fix packs, subscribe to My Support e-mail updates on the DB2 Technical Support Web site at: http://www.ibm.com/software/data/db2/udb/support

Applying test fixes

A test fix is a temporary fix that is supplied to specific customers for testing in response to a reported problem. A Readme accompanies every test fix. The test fix Readme provides instructions for installing and uninstalling the test fix, as well as a list of the APARs (if any) included in the test fix.

Each test fix has specific prerequisites. Refer to the Readme that accompanies the test fix for details.

There are two types of test fixes:
- A test fix for an individual DB2 product. These test fixes can be applied on an existing installation of the product, or can be used to perform a full product installation where there is no existing DB2 installation.
- Universal test fixes (Linux and UNIX only). A universal test fix services installations where more than one DB2 product has been installed.

If national languages have been installed, you might also require a separate national language test fix. The national language test fix can only be applied if it is at the same test fix level as the installed DB2 product. If you are applying a universal test fix, you must apply both the universal test fix and the national language test fix to update the DB2 products.

Obtain the test fix from DB2 Customer Support and follow the instructions in the Readme with respect to installing, testing and removing (if necessary) the test fix.

When installing a test fix in a multi-partition database partition environment, the system must be offline and all computers participating in the instance must be upgraded to the same test fix level.

Appendix A. Overview of the DB2 technical information

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
  - Topics (Task, concept and reference topics)
  - Help for DB2 tools
  - Sample programs
  - Tutorials
- DB2 books
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF DVD)
  - printed books
- Command line help
  - Command help
  - Message help

Note: The DB2 Information Center topics are updated more frequently than either the PDF or the hard-copy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 Information Center at ibm.com®.

You can access additional DB2 technical information such as technotes, white papers, and IBM Redbooks publications online at ibm.com. Access the DB2 Information Management software library site at http://www.ibm.com/software/data/sw-library/.

Documentation feedback

We value your feedback on the DB2 documentation. If you have suggestions for how to improve the DB2 documentation, send an email to db2docs@ca.ibm.com. The DB2 documentation team reads all of your feedback, but cannot respond to you directly. Provide specific examples wherever possible so that we can better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use this email address to contact DB2 Customer Support. If you have a DB2 technical issue that the documentation does not resolve, contact your local IBM service center for assistance.

DB2 technical library in hardcopy or PDF format

The following tables describe the DB2 library available from the IBM Publications Center at www.ibm.com/shop/publications/order. English DB2 Version 9.5 manuals in PDF format and translated versions can be downloaded from www.ibm.com/support/docview.wss?rs=71&uid=swg2700947.

Although the tables identify books available in print, the books might not be available in your country or region.

The form number increases each time a manual is updated. Ensure that you are reading the most recent version of the manuals, as listed below.

Note: The DB2 Information Center is updated more frequently than either the PDF or the hard-copy books.

Table 6. DB2 technical information

Name	Form Number	Available in print
Administrative API Reference	SC23-5842-01	Yes
Administrative Routines and Views	SC23-5843-01	No
Call Level Interface Guide and Reference, Volume 1	SC23-5844-01	Yes
Call Level Interface Guide and Reference, Volume 2	SC23-5845-01	Yes
Command Reference	SC23-5846-01	Yes
Data Movement Utilities Guide and Reference	SC23-5847-01	Yes
Data Recovery and High Availability Guide and Reference	SC23-5848-01	Yes
Data Servers, Databases, and Database Objects Guide	SC23-5849-01	Yes
Database Security Guide	SC23-5850-01	Yes
Developing ADO.NET and OLE DB Applications	SC23-5851-01	Yes
Developing Embedded SQL Applications	SC23-5852-01	Yes
Developing Java Applications	SC23-5853-01	Yes
Developing Perl and PHP Applications	SC23-5854-01	No
Developing User-defined Routines (SQL and External)	SC23-5855-01	Yes
Getting Started with Database Application Development	GC23-5856-01	Yes
Getting Started with DB2 installation and administration on Linux and Windows	GC23-5857-01	Yes
Internationalization Guide	SC23-5858-01	Yes
Message Reference, Volume 1	GI11-7855-00	No
Message Reference, Volume 2	GI11-7856-00	No
Migration Guide	GC23-5859-01	Yes
Net Search Extender Administration and User's Guide	SC23-8509-01	Yes
Partitioning and Clustering Guide	SC23-5860-01	Yes
Query Patroller Administration and User's Guide	SC23-8507-00	Yes
Quick Beginnings for IBM Data Server Clients	GC23-5863-01	No

Table 6. DB2 technical information (continued)

Name	Form Number	Available in print
Quick Beginnings for DB2 Servers	GC23-5864-01	Yes
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC23-8508-01	Yes
SQL Reference, Volume 1	SC23-5861-01	Yes
SQL Reference, Volume 2	SC23-5862-01	Yes
System Monitor Guide and Reference	SC23-5865-01	Yes
Troubleshooting Guide	GI11-7857-01	No
Tuning Database Performance	SC23-5867-01	Yes
Visual Explain Tutorial	SC23-5868-00	No
What's New	SC23-5869-01	Yes
Workload Manager Guide and Reference	SC23-5870-01	Yes
pureXML Guide	SC23-5871-01	Yes
XQuery Reference	SC23-5872-01	No

Table 7. DB2 Connect-specific technical information

Name	Form Number	Available in print
Quick Beginnings for DB2 Connect Personal Edition	GC23-5839-01	Yes
Quick Beginnings for DB2 Connect Servers	GC23-5840-01	Yes
DB2 Connect User's Guide	SC23-5841-01	Yes

Table 8. Information Integration technical information

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

Ordering printed DB2 books

If you require printed DB2 books, you can buy them online in many but not all countries or regions. You can always order printed DB2 books from your local IBM representative. Keep in mind that some softcopy books on the *DB2 PDF Documentation* DVD are unavailable in print. For example, neither volume of the *DB2 Message Reference* is available as a printed book.

Printed versions of many of the DB2 books available on the DB2 PDF Documentation DVD can be ordered for a fee from IBM. Depending on where you are placing your order from, you may be able to order books online, from the IBM Publications Center. If online ordering is not available in your country or region, you can always order printed DB2 books from your local IBM representative. Note that not all books on the DB2 PDF Documentation DVD are available in print.

Note: The most up-to-date and complete DB2 documentation is maintained in the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2luw/v9r5.

To order printed DB2 books:

- To find out whether you can order printed DB2 books online in your country or region, check the IBM Publications Center at http://www.ibm.com/shop/publications/order. You must select a country, region, or language to access publication ordering information and then follow the ordering instructions for your location.
- To order printed DB2 books from your local IBM representative:
  1. Locate the contact information for your local representative from one of the following Web sites:
     - The IBM directory of world wide contacts at www.ibm.com/planetwide
     - The IBM Publications Web site at http://www.ibm.com/shop/publications/order. You will need to select your country, region, or language to the access appropriate publications home page for your location. From this page, follow the "About this site" link.
  2. When you call, specify that you want to order a DB2 publication.
  3. Provide your representative with the titles and form numbers of the books that you want to order. For titles and form numbers, see "DB2 technical library in hardcopy or PDF format" on page 103.

Displaying SQL state help from the command line processor

DB2 returns an SQLSTATE value for conditions that could be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

To invoke SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.

For example, `? 08003` displays help for the 08003 SQL state, and `? 08` displays help for the 08 class code.

Accessing different versions of the DB2 Information Center

For DB2 Version 9.5 topics, the DB2 Information Center URL is http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/

For DB2 Version 9 topics, the DB2 Information Center URL is http://publib.boulder.ibm.com/infocenter/db2luw/v9/

For DB2 Version 8 topics, go to the Version 8 Information Center URL at: http://publib.boulder.ibm.com/infocenter/db2luw/v8/

Displaying topics in your preferred language in the DB2 Information Center

The DB2 Information Center attempts to display topics in the language specified in your browser preferences. If a topic has not been translated into your preferred language, the DB2 Information Center displays the topic in English.

- To display topics in your preferred language in the Internet Explorer browser:
  1. In Internet Explorer, click the **Tools** —> **Internet Options** —> **Languages...** button. The Language Preferences window opens.
  2. Ensure your preferred language is specified as the first entry in the list of languages.
     - To add a new language to the list, click the **Add...** button.

       Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.
     - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
  3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.
- To display topics in your preferred language in a Firefox or Mozilla browser:
  1. Select the button in the **Languages** section of the **Tools** —> **Options** —> **Advanced** dialog. The Languages panel is displayed in the Preferences window.
  2. Ensure your preferred language is specified as the first entry in the list of languages.
     - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
     - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
  3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

On some browser and operating system combinations, you might have to also change the regional settings of your operating system to the locale and language of your choice.

Updating the DB2 Information Center installed on your computer or intranet server

If you have installed the DB2 Information Center locally, you can obtain and install documentation updates from IBM.

Updating your locally-installed DB2 Information Center requires that you:

1. Stop the DB2 Information Center on your computer, and restart the Information Center in stand-alone mode. Running the Information Center in stand-alone mode prevents other users on your network from accessing the Information Center, and allows you to apply updates. Non-Administrative and Non-Root DB2 Information Centers always run in stand-alone mode. .
2. Use the Update feature to see what updates are available. If there are updates that you would like to install, you can use the Update feature to obtain and install them

   Note: If your environment requires installing the DB2 Information Center updates on a machine that is not connected to the internet, you have to mirror the update site to a local file system using a machine that is connected to the internet and has the DB2 Information Center installed. If many users on your network will be installing the documentation updates, you can reduce the time required for individuals to perform the updates by also mirroring the update site locally and creating a proxy for the update site.
   If update packages are available, use the Update feature to get the packages. However, the Update feature is only available in stand-alone mode.
3. Stop the stand-alone Information Center, and restart the DB2 Information Center on your computer.

Note: On Windows Vista, the commands listed below must be run as an administrator. To launch a command prompt or graphical tool with full administrator privileges, right-click on the shortcut and then select **Run as administrator**.

To update the DB2 Information Center installed on your computer or intranet server:

1. Stop the DB2 Information Center.
   - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Stop**.
   - On Linux, enter the following command:
     ```
     /etc/init.d/db2icdv95 stop
     ```
2. Start the Information Center in stand-alone mode.
   - On Windows:
     a. Open a command window.
     b. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the <Program Files>\IBM\DB2 Information Center\Version 9.5 directory, where `<Program Files>` represents the location of the Program Files directory.
     c. Navigate from the installation directory to the doc\bin directory.
     d. Run the help_start.bat file:
        ```
        help_start.bat
        ```
   - On Linux:

a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the /opt/ibm/db2ic/V9.5 directory.
b. Navigate from the installation directory to the doc/bin directory.
c. Run the help_start script:

```
help_start
```

The systems default Web browser launches to display the stand-alone Information Center.

3. Click the **Update** button ( ). On the right hand panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the installation process, check the selections you want to install, then click **Install Updates**.
5. After the installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center:
   - On Windows, navigate to the installation directory's doc\bin directory, and run the help_end.bat file:

     ```
     help_end.bat
     ```

     Note: The help_end batch file contains the commands required to safely terminate the processes that were started with the help_start batch file. Do not use `Ctrl-C` or any other method to terminate help_start.bat.
   - On Linux, navigate to the installation directory's doc/bin directory, and run the help_end script:

     ```
     help_end
     ```

     Note: The help_end script contains the commands required to safely terminate the processes that were started with the help_start script. Do not use any other method to terminate the help_start script.
7. Restart the DB2 Information Center.
   - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Start**.
   - On Linux, enter the following command:

     ```
     /etc/init.d/db2icdv95 start
     ```

The updated DB2 Information Center displays the new and updated topics.

DB2 tutorials

The DB2 tutorials help you learn about various aspects of DB2 products. Lessons provide step-by-step instructions.

Before you begin

You can view the XHTML version of the tutorial from the Information Center at http://publib.boulder.ibm.com/infocenter/db2help/.

Some lessons use sample data or code. See the tutorial for a description of any prerequisites for its specific tasks.

DB2 tutorials

To view the tutorial, click on the title.

"pureXML™" in *pureXML Guide*
: Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

"Visual Explain" in *Visual Explain Tutorial*
: Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 products.

DB2 documentation
: Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

DB2 Technical Support Web site
: Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs or bug fixes), fix packs, and other resources. You can search through this knowledge base to find possible solutions to your problems.

  Access the DB2 Technical Support Web site at http://www.ibm.com/software/data/db2/udb/support.html

Terms and Conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal use: You may reproduce these Publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these Publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

Appendix B. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This document may provide links or references to non-IBM Web sites and resources. IBM makes no representations, warranties, or other commitments whatsoever about any non-IBM Web sites or third-party resources that may be referenced, accessible from, or linked from this document. A link to a non-IBM Web site does not mean that IBM endorses the content or use of such Web site or its owner. In addition, IBM is not a party to or responsible for any transactions you may enter into with third parties, even if you learn of such parties (or use a link to such parties) from an IBM site. Accordingly, you acknowledge and agree that IBM is not responsible for the availability of such external sites or resources, and is not responsible or liable for any content, services, products, or other materials on or available from those sites or resources. Any software provided by third parties is subject to the terms and conditions of the license that accompanies that software.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _*enter the year or years*_. All rights reserved.

Trademarks

The following terms are trademarks or registered trademarks of the International Business Machines Corporation in the United States, other countries, or both.

pureXML	OS/390
DB2 Connect	Passport Advantage
Distributed Relational Database Architecture	z/OS
Redbooks	System i
IBM	DB2
zSeries	AIX
Tivoli	DRDA
ibm.com	i5/OS
iSeries	

The following terms are trademarks or registered trademarks of other companies

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Windows is a registered trademark of Microsoft Corporation in the United States, other countries, or both.

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

Index

Printed in USA

GI11-7857-01

Spine information:

DB2 Version 9.5 for Linux, UNIX, and Windows

Troubleshooting Guide