DB2

IBM

DB2 Version 9
for Linux, UNIX, and Windows

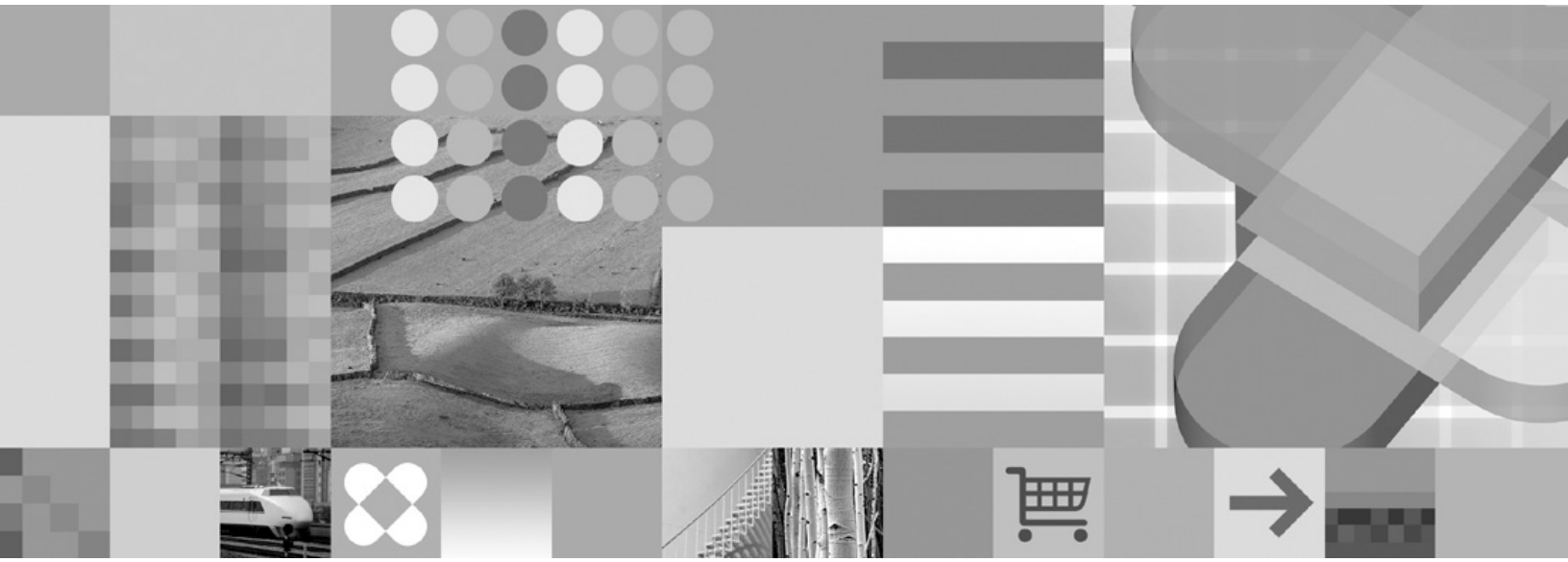
Troubleshooting Guide

GC10-4240-00

DB2®

IBM

DB2 Version 9
for Linux, UNIX, and Windows



Troubleshooting Guide

GC10-4240-00

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

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).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Learning more

Introduction to problem determination

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 may 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 may 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.

Related concepts:

• "Problem determination resources" on page 76

About first failure data capture(FFDC)

First failure data capture information

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

For more details about specific types of FFDC, refer to the related links listed below.

Related concepts:

• "Dump Files" on page 13
• "Trap files" on page 14

Related reference:

• "General First Failure Data Capture (FFDC)" on page 3

General First Failure Data Capture (FFDC)

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 problem determination and is intended for IBM® customer 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 customer 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.

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 customer 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.
- The core file is a binary file that contains information similar to the trap files produced by DB2 database products. Core files can also contain the entire memory image of the terminated process.

Related concepts:

- "Administration notification log" on page 5
- "db2cos (callout script) output files" on page 11
- "Dump Files" on page 13
- "Interpreting administration notification log file entries" on page 6
- "Trap files" on page 14

About administration notification log files

Administration notification log

When significant events occur, the DB2 database manager writes information to the administration notification log. For example, it records the status of DB2 utilities (REORG, BACKUP, RECOVERY, ROLL-FORWARD), high-level application issues, licensing activity, log file paths and storage problems, monitoring and indexing activities, table space problems, and so on. This information is intended for use by database and system administrators.

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. This detailed diagnostic information is recorded in the db2diag.log text log file, not in the administration log. Diagnostic information is used for problem determination and is intended for DB2 customer support. The level of detail is determined by the DIAGLEVEL configuration parameter.

Related concepts:

- "Interpreting administration notification log file entries" on page 6
- "Interpreting diagnostic log file entries" on page 9
- "Interpreting the db2diag.log file informational record" on page 8

Related tasks:

- "Setting the diagnostic log file error capture level" on page 8
- "Setting the error capture level for the administration notification log file" on page 6

Related reference:

- "notifylevel - Notify level configuration parameter" in *Performance Guide*
- "diaglevel - Diagnostic error capture level configuration parameter" in *Performance Guide*

Setting the error capture level for the administration notification log file

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

```
DB2 GET DBM CFG
```

Look for the following variable:

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

To alter the setting, use the command:

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

where X is the desired notification level.

Related reference:

- "notifylevel - Notify level configuration parameter" in *Performance Guide*

Interpreting administration notification log file entries

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

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

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

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

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

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

```
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 partition generating the message. (In a non-partitioned database, the value is "000".)

4. The DB2 component that is writing the message. For messages written by user applications using the `db2AdminMsgWrite` API, the component will read "User Application".
5. Identification of the application for which the process is working. In this example, the process generating the message is working on behalf of an application with the ID *LOCAL.DB2.020205091435.

   To identify more about a particular application ID, either:

   - Use the **db2 list applications** command on a DB2 server or **db2 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 **db2 get snapshot for application** command to view a list of application IDs.
6. The DB2 component that is writing the message. For messages written by user applications using the `db2AdminMsgWrite` API, the component will read "User Application".
7. The name of the function that is providing the message. This function operates within the DB2 subcomponent that is writing the message. For messages written by user applications using the `db2AdminMsgWrite` API, the function will read "User Function".

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

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

   b Buffer pools

   c Communication between clients and servers

   d Data management

   e Engine processes

   o Operating system calls (such as opening and closing files)

   p Data protection (such as locking and logging)

   r Relational database services

   s Sorting

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

Related concepts:

- "Administration notification log" on page 5
- "First failure data capture information" on page 3

Related reference:

- "notifylevel - Notify level configuration parameter" in *Performance Guide*

About DB2 diagnostic log (db2diag.log) files

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 problem determination and is 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:

```
DB2 GET DBM CFG
```

Look for the following variable:

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

To alter the setting, use the command:

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

where `X` is the desired notification level.

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

Related concepts:

- "First failure data capture information" on page 3

Related reference:

- "diaglevel - Diagnostic error capture level configuration parameter" in *Performance Guide*

Interpreting the db2diag.log file informational record

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

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

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

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

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

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

```
2006-02-13-14.34.35.965000-300 1   I17502H435 2       LEVEL: Error 3
PID     : 940 4                TID  : 660 5     PROC : db2syscs.exe 6
INSTANCE: DB2 7                NODE : 000 8      DB   : SAMPLE 9
APPHDL  : 0-1433 10               APPID: *LOCAL.DB2.050120082811 11
FUNCTION: 12 DB2 UDB, data protection, sqlpsize, probe:20
RETCODE : 13 ZRC=0x860F000A=-2045837302=SQLO_FNEX "File not found."
           DIA8411C A file "" could not be found.
```

Legend:

1. A timestamp and timezone for the message.
2. The record ID field. The db2diag.log's recordID specifies the file offset at which the current message is being logged (for example, ″17502″) and the message length (for example, ″435″) 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 partition generating the message. (In a non-partitioned database, the value is ″000″.)
9. The database name
10. The application handle. This value aligns with that used in db2pd output and lock dump files. It consists of the coordinator partition number followed by the coordinator index number, separated by a dash.
11. Identification of the application for which the process is working. In this example, the process generating the message is working on behalf of an application with the ID *LOCAL.DB2.050120082811.

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

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

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

To identify more about a particular application ID, either:

- Use the **db2 list applications** command on a DB2 server or **db2 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 **db2 get snapshot for application** command to view a list of application IDs.
- Use the **db2pd -applications -db sample** command.

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

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

Note: Timestamps in the db2diag.log contain a time zone. 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).

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

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

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

- `<source>` Indicates the origin of the logged error. 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. 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

Related concepts:

- "Interpreting the db2diag.log file informational record" on page 8

db2cos (callout script) output files

The db2cos script is invoked by default when the database manager cannot continue processing due to a panic, trap, segmentation violation or exception. The default db2cos script will invoke db2pd commands to collect information in an unlatched manner.

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. Generated signals, on the other hand, 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 db2cosXXXYYY.ZZZ, 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    ................
```

Related concepts:

- "First failure data capture information" on page 3
- "Trap files" on page 14

Related reference:

- "db2pd - Monitor and troubleshoot DB2 database command" in *Command Reference*
- "db2pdcfg - Configure DB2 database for problem determination behavior command" in *Command Reference*

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:

```
2006-03-16-11.53.18.001160    Instance:payroll   Node:000
PID:44829(db2agent (SAMPLE))     Appid:*LOCAL.payroll.970317140834
relation_data_serv          sqlrerlg    Probe:17   Database:SAMPLE
DIA9999E An internal return code occurred. Report the following : "0xFFFFE101".
Dump File:  /home/db2/sqllib/db2dump/56772.000 Data : SECTION STMT  1
```

Legend:

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

Notes:

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

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

Related concepts:

- "First failure data capture information" on page 3

About trap files

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 traceback.″ 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 Linux and UNIX, the first letter in the file name is ″t″, followed by a process identifier (PID). The file extension is the partition number (000 on single partition databases).

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

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 hexadecimal format, followed by the partition number (0 in a single partition database).

Examples:

- t56772.000 is a trap file for the process with pid 56772.
- t56772.010 is a trap file for the process with pid 56772. It was created by a DB2 process running on partition 10.

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.

Related concepts:

- "First failure data capture information" on page 3
- "System core files (UNIX)" on page 16

Related tasks:

- "Monitoring and troubleshooting using db2pd" on page 37

Related reference:

- "db2pd - Monitor and troubleshoot DB2 database command" in *Command Reference*

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
```

Related concepts:

- "First failure data capture information" on page 3
- "Trap files" on page 14

Related reference:

- "db2xprt - Format trap file command" in *Command Reference*

About platform-specific error logs

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 DB2 files available. Some of this information is contained in logs and traces within the following areas:

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

Based on your operating environment, there may be more places outside of what has been described here, so be aware of all of the potential areas you may 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 may be available for piece of hardware in your environment. An example of such a case is when a bad page, or a corruption of some type is reported by DB2. Usually this is reported due to a disk problem, in which case the hardware diagnostics would need to be investigated. Please refer to your hardware documentation and support material to determine what further information is available.

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

Related concepts:

- "First failure data capture information" on page 3

System core files (UNIX)

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

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

Related concepts:

- "First failure data capture information" on page 3

Accessing system core file information (UNIX)

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

Prerequisites:

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

Procedure:

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

1. Enter the following command from a UNIX command prompt:

   ```
   dbx program_name core_filename
   ```

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

   The *core_filename* parameter is optional. If you do not specify it, the default name ″core″ is used.
2. To obtain symbolic information, compile the application using the ″-g″ option.
3. To end the **dbx** command, type **quit** at the dbx prompt.
4. For the HP-UX operating system, use the **xdb** command for similar function.
5. On AIX, ensure that the full core option has been enabled using the **chdev** command or smitty.
6. The **dbx** command provides much more function than is described in this section. To find out more, enter `man dbx` from a UNIX command prompt.

Example of the dbx command:

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

1. At a command prompt, enter:

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

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

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

In this example, the error occurred at line 136 of freeSegments, which was called from line 96 in main.c.

4. To end the **dbx** command, type **quit** at the dbx prompt.

Related concepts:

- "Diagnostic tools (Linux and UNIX)" on page 71
- "System core files (UNIX)" on page 16

Accessing event logs (Windows)

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

- System
- Application
- Security

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

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

Related concepts:

- "First failure data capture information" on page 3

Related tasks:

- "Exporting event logs (Windows)" on page 18

Exporting event logs (Windows)

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

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

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

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

Related concepts:

- "First failure data capture information" on page 3

Related tasks:

- "Accessing event logs (Windows)" on page 18

Accessing the Dr. Watson log file (Windows)

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

Related concepts:

- "First failure data capture information" on page 3

Related reference:

- "Diagnostic tools (Windows)" on page 70

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 or NIS servers, 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.

Related reference:

- "db2pd - Monitor and troubleshoot DB2 database command" in *Command Reference*
- "db2support - Problem analysis and environment collection tool command" in *Command Reference*

Chapter 2. Troubleshooting DB2

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.

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.

Troubleshooting partitioned database environments

Issuing commands in a partitioned database environment

In a partitioned database environment, you might want to issue commands to be run on computers in the instance, or on database partition servers (nodes). You can do so using the **rah** command or the **db2_all** command. The **rah** command allows you to issue commands that you want to run at computers in the instance. If you want the commands to run at database partition servers in the instance, you run the **db2_all** command. This section provides an overview of these commands. The information that follows applies to partitioned database environments only.

Notes:

1. On Linux and UNIX platforms, your login shell can be a Korn shell or any other shell; however, there are differences in the way the different shells handle commands containing special characters.
2. Also, on Linux and UNIX platforms, rah uses the remote shell program specified by the DB2RSHCMD registry variable. You can select between the two remote shell programs: ssh (for additional security), or rsh (or remsh for HP-UX). The ssh remote shell program is used to prevent the transmission of passwords in clear text in UNIX operating system environments. If this registry variable is not set, rsh (or remsh for HP-UX) is used.
3. On Windows, to run the **rah** command or the **db2_all** command, you must be logged on with a user account that is a member of the Administrators group.

To determine the scope of a command, refer to the *Command Reference*, which indicates whether a command runs on a single database partition server, or on all of them. If the command runs on one database partition server and you want it to run on all of them, use **db2_all**. The exception is the **db2trc** command, which runs on all the logical nodes (database partition servers) on a computer. If you want to run **db2trc** on all logical nodes on all computers, use **rah**.

Related concepts:

- "rah and db2_all commands overview" in *Administration Guide: Implementation*
- "Specifying the rah and db2_all commands" in *Administration Guide: Implementation*

Related reference:

- "rah and db2_all command descriptions" in *Administration Guide: Implementation*

Chapter 3. Troubleshooting DB2 Connect

Problem determination

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

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

Related concepts:

- "Diagnostic tools" on page 28
- "Gathering relevant information" on page 25
- "Initial connection is not successful" on page 26
- "Problems encountered after an initial connection" on page 27
- "Trace utility" on page 52

Gathering relevant information

Problem determination includes narrowing the scope of the problem and investigating the possible causes. The proper starting point is to gather the relevant information and determine what you know, what data has not been gathered, and what paths you can eliminate. At a minimum answer the following questions.

- 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 client and DB2 Connect workstation, DB2 Connect workstation and host or iSeries™ database server, all clients or one client?
- What can you determine by the content of the message and the tokens returned in the message?
- Will using diagnostic tools provide any assistance at this time?
- Are other machines performing similar tasks working correctly?
- If this is a remote task, is it successful if performed locally?

Related concepts:

- "Diagnostic tools" on page 28
- "Problem determination" on page 25

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 iSeries database server and workstation communications configured properly?*
   - There are three configurations that must be considered:
     a. The host or iSeries database server configuration identifies the application requester to the server. The host or iSeries 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 iSeries 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 iSeries 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 iSeries server database management system to use the host or iSeries 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 iSeries database server, are you unsuccessful?*
   - Did you follow the procedure to bind the CLP to the host or iSeries database server?

Related concepts:

- "Problem determination" on page 25
- "Problems encountered after an initial connection" on page 27

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 PTF 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 PTF level and load that level *after* installing the feature.
2. *Has this error occurred before?*
   - Is there any documented resolution to previous error conditions?
   - Who were the participants and can they provide insight into 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 previously not able 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.

Note: DB2 Connect does not support the PING command when issued from a Version 7 client through a Version 9 DB2 Connect server.

Related concepts:

- "Problem determination" on page 25
- "Trace utility" on page 52

Diagnostic tools

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

- The first failure service log, where diagnostic information is consolidated and stored in a readable format, is stored in the administration notification log.
- Both logs are found in the path specified:

  This file is located in `/u/db2/sqllib/db2dump/notifyloglevel.nfy` on UNIX systems, where `db2` represents the instance name.

  This file is located in `x:\sqllib\db2\db2diag.log` on Windows systems, where `x:` represents the logical drive, and `db2` represents the instance name.
- For Windows operating systems, you can use the Event Viewer to view the administration notification log.
- The trace utility
- 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.

Related concepts:

- "DB2 Connect performance troubleshooting" in *DB2 Connect User's Guide*
- "Trace utility" on page 52

Chapter 4. Tools for troubleshooting

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 environments, the file is created in the current directory. For multiple-partition database environments, the file is created under a subdirectory in the diagnostic directory. The subdirectory is called DART####, where #### is the 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 may 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.

Related reference:

- "db2dart - Database analysis and reporting tool command" in *Command Reference*

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.

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

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

If there are several databases in the instance, and you wish 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=940 -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     : 940                  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 the 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

Related concepts:

- “First failure data capture information” on page 3

Related reference:

- “db2diag - db2diag.log analysis tool command” in *Command Reference*

Displaying and altering the Global Registry (UNIX) using db2greg

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

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

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

Related reference:

- “Contacting IBM” on page 95

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
```

Related concepts:

- “What's new for V9.1: Coexistence of multiple DB2 versions and fix packs enhancements (Linux and UNIX)” in *What's New*
- “What's new for V9.1: Coexistence of multiple DB2 versions and fix packs now supported (Windows)” in *What's New*

Related tasks:

- “Listing DB2 products installed on your system (Linux and UNIX)” on page 35

Related reference:

- "db2level - Show DB2 service level command" in *Command Reference*
- "db2ls - List installed DB2 products and features command" in *Command Reference*
- "db2support - Problem analysis and environment collection tool command" in *Command Reference*

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 may be easier to debug the problem on a test system that is identical to the production system.

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

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

Using db2look to mimic the tables in a database:

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

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

Note: If you want the DDL for the user-defined spaces, database partition groups and buffer pools to be produced as well, add the**-l** flag after **-e** in the command above. The default 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 wish 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 may 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″.

Related concepts:

- “Statistics for modeling production databases” in *Performance Guide*

Related reference:

- “db2look - DB2 statistics and DDL extraction tool command” in *Command Reference*

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

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

Prerequisites:

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

Restrictions:

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. Any existing scripts containing a native installation utility that you use to interface and query with DB2 installations will need to change.

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

Procedure:

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
- Installation date listing when the DB2 product was last modified.

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

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

where

- *q* specifies you are querying a product or feature. This parameter is mandatory. If a DB2 Version 8 product is queried, a blank value is returned.
- *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, 9.1.0.0
  - 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.

Related concepts:

- "Multiple DB2 copies on the same computer (Linux and UNIX)" in *Installation and Configuration Supplement*

Related tasks:

- "Installing a DB2 product using the db2_install or doce_install command (Linux and UNIX)" in *Installation and Configuration Supplement*
- "Removing DB2 products using the db2_deinstall or doce_deinstall command (Linux and UNIX)" in *Quick Beginnings for DB2 Servers*
- "Using the Default DB2 Selection wizard (Windows)" in *Quick Beginnings for DB2 Servers*

Related reference:

- "db2ls - List installed DB2 products and features command" in *Command Reference*
- "Multiple DB2 copies roadmap" in *Administration Guide: Implementation*

Monitoring and troubleshooting using db2pd

The **db2pd** tool is a problem determination tool that contains quick and immediate information from the DB2 memory sets.

The tool collects this 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 problem determination. 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** could be used to expedite problem determination.

Scenario 1: Catching a lock timeout using the **db2pdcfg -catch** option

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

```
db2pdcfg -catch locktimeout count=1
Error Catch #1
Sqlcode: 0
ReasonCode: 0
ZRC: -2146435004
ECF: 0
Component ID: 0
LockName: Not Set
LockType: Not Set
Current Count: 0
Max Count: 1
Bitmap: 0x4A1
Action: Error code catch flag enabled
Action: Execute sqllib/db2cos callout script
Action: Produce stack trace in db2diag.log
```

Reproduce a lock timeout error, for example:

```
SQL0911N  The current transaction has been rolled back because of a deadlock or timeout.
Reason code "68".  SQLSTATE=40001
```

A corresponding message will appear in the db2diag.log:

```
2006-02-17-01.28.41.803000-300 I22649H285         LEVEL: Event
PID     : 940                  TID  : 3136        PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000
FUNCTION: DB2 UDB, RAS/PD component, pdErrorCatch, probe:30
START   : Error catch set for ZRC -2146435004

...

2006-02-17-01.40.04.091000-300 I23330H389         LEVEL: Event
PID     : 940                  TID  : 2136        PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-251                APPID: *LOCAL.DB2.060217064002
FUNCTION: DB2 UDB, lock manager, sqlplnfd, probe:999
DATA #1 : preformatted
Caught rc -2146435004.  Dumping stack trace.

2006-02-17-01.40.04.106000-300 I23721H416         LEVEL: Event
PID     : 940                  TID  : 2136        PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-251                APPID: *LOCAL.DB2.060217064002
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:10
START   : Invoking C:\PROGRA~1\IBM\SQLLIB\bin\db2cos.bat from lock manager sqlplnfd

...

2006-02-17-01.40.07.715000-300 I25489H399         LEVEL: Event
PID     : 940                  TID  : 2136        PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-251                APPID: *LOCAL.DB2.060217064002
FUNCTION: DB2 UDB, trace services, pdInvokeCalloutScript, probe:20
STOP    : Completed invoking C:\PROGRA~1\IBM\SQLLIB\bin\db2cos.bat
```

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:
<snip>
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:06:53
Locks:
Address TranHdl Lockname Type Mode Sts Owner Dur HldCnt Att Rlse
0x402C6B30 3 00020003000000040000000052 Row ..X W* 3 1 0 0 0x40
```

Just look for the "W*" as this is the lock that experienced the timeout. You can map this to a transaction, application, agent, and even SQL statement with all of the output provided by the **db2pd** command. 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's what you need.

Scenario 2: 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 statements.

```
db2pd -app -dyn

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

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

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

Scenario 3: 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 4: 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 an explicit 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 UsablePgs column and comparing it to the TotalPages value.

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 output from **db2pd -agent** will show the number of rows written as verification.

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

State       ClientPid  Userid   ClientNm Rowsread   Rowswrtn   LkTmOt
Inst-Active 26377      db2inst1 db2bp    22         9588       NotSet
```

The steps are slightly different if you suspect that an implicit 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 <AppHandl><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 5: 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 6: 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 7: 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
0x0000000023001BF58 0x000001B58000 0x00000000 1000  1000   S0000002.LOG
0x0000000023001BE98 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 8: 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 9: Generating stack traces

The **db2pd -stack all** command can be used to produce stack traces for the database system. 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 process by issuing the command **db2pd -stack <pid>**. For example:

```
db2pd -stack 1335470

  Attempting to dump stack trace for pid 1335470.
  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:

```
db2pd -stack all

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

If you are using a multi-partitioned 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**.

Related reference:

- "db2pd - Monitor and troubleshoot DB2 database command" in *Command Reference*

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.

Note: The db2support utility should be run by a user with SYSADM authority, such as an instance owner, so that the utility can collect all of the necessary information without an error. If a user without SYSADM authority runs **db2support**, SQL errors (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 failure data capture (FFDC) diagnostic files are produced automatically.

Executing **db2support -h** brings up the complete list of command options. 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
- Core files (**-a** option collects all core files, **-r** option only collects the most recent core file)
- 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

Related reference:

- "db2support - Problem analysis and environment collection tool command" in *Command Reference*

Traces

Basic trace diagnostics

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

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

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

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

- Simple, step by step procedures
- An explanation of where each trace is to be taken
- An explanation of what should be traced
- An explanation of why the trace is requested
- Backout procedures (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.

Related concepts:

- "Obtaining a DB2 trace using db2trc" on page 49
- "DRDA trace files" on page 52

DB2 traces

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 may impact your system's performance.

In general, DB2 Support and development teams use DB2 traces for problem determination. 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.

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

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

If tracing an operation that is relatively short (such as a database connection), a size of approximately 8MB 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 may be required.

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

1. On multiple database partition systems, you must run a trace for each physical (as opposed to logical) database partition.
2. On 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 regardless of the size specified. To effectively run a trace on these platforms, turn the trace on before starting the instance and "clear" it as necessary afterwards.

Related concepts:

- "Dumping a DB2 trace file" on page 50
- "Formatting a DB2 trace file" on page 50

Related reference:

- "db2trc - Trace command" in *Command Reference*

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
```

Related concepts:

- "Formatting a DB2 trace file" on page 50
- "Obtaining a DB2 trace using db2trc" on page 49

Related reference:

- "db2trc - Trace command" in *Command Reference*

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 may sometimes be asked to format the dump file into ASCII format before sending it. This is accomplished via the **flw** and **fmt** options. You must provide the 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.

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 node number. This does not occur on Windows platforms. You have to dump the trace manually in those situations.

To summarize, the following is an example of 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
```

Related concepts:

- “Dumping a DB2 trace file” on page 50
- “Obtaining a DB2 trace using db2trc” on page 49

Related reference:

- “db2trc - Trace command” in *Command Reference*

DRDA traces

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 iSeries database server).

Related concepts:

- "DB2 Connect and DRDA" in *DB2 Connect User's Guide*
- "Trace utility" on page 52

Related reference:

- "db2drdat - DRDA trace command" in *Command Reference*

Trace utility

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

As a database administrator (or application developer), you 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. For example, if you issue a `CONNECT TO` database statement for a host or iSeries database server, but the command fails and you receive an unsuccessful return code. If you understand exactly what information was conveyed to the host or iSeries database server management system, you 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 iSeries database server management system. Data sent to the host or iSeries database server is labeled SEND BUFFER and data received from the host or iSeries database server is labeled RECEIVE BUFFER.

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

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

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

- A byte count.
- Columns 2 and 3 represent the DRDA 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.

Related concepts:

- "Trace output" on page 53
- "Trace output file analysis" on page 54

Related reference:

- "db2drdat - DRDA trace command" in *Command Reference*

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.

Notes:

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.

Related concepts:

- "Trace output file analysis" on page 54
- "Trace utility" on page 52

Related reference:

- "db2drdat - DRDA trace command" in *Command Reference*

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 iSeries database server CCSID(s)
- The host or iSeries 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 iSeries 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 iSeries 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 iSeries 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 database 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 iSeries 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 iSeries 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 iSeries database server, SQLCODE -332 will be returned to the user with the source and target code pages. If the host or iSeries 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 UDB for iSeries.

Related concepts:

- "Trace output" on page 53
- "Trace utility" on page 52

Related reference:

- "db2drdat - DRDA trace command" in *Command Reference*
- "Subsequent buffer information for DRDA traces" on page 61
- "Trace output file samples" on page 55

Trace output file samples

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

Figure 1 uses DB2 Connect Enterprise Edition Version 9.1 and DB2 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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 1 of 20)

```
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                        @@@@@@
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 2 of 20)

```
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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 3 of 20)

```
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..
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 4 of 20)

```
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 1. Example of Trace Output (TCP/IP connection) (Part 5 of 20)

```
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  4040404040400000C 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.
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 6 of 20)

```
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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 7 of 20)

```
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 D5404000006212524   .........@@..!%$  ...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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 8 of 20)

```
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 1. Example of Trace Output (TCP/IP connection) (Part 9 of 20)

```
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               ........ .        ..}.......
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 10 of 20)

```
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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 11 of 20)

```
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            ........$..       ..}........
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 12 of 20)

```
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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 13 of 20)

```
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  @@@@AAAAAAHLV....      ......<.....
  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.       ..._`./.%..
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 14 of 20)

```
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 1. Example of Trace Output (TCP/IP connection) (Part 15 of 20)

```
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...            .<....
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 16 of 20)

```
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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 17 of 20)

```
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               ........ .        ..}.......
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 18 of 20)

```
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
```

Figure 1. Example of Trace Output (TCP/IP connection) (Part 19 of 20)

```
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 1. Example of Trace Output (TCP/IP connection) (Part 20 of 20)

Related concepts:

- "Trace output file analysis" on page 54

Related reference:

- "Subsequent buffer information for DRDA traces" on page 61

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 iSeries database server management system to commit the current unit of work. The fourth buffer is received from the host or iSeries 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 1 on page 55 shows an example of a receive buffer containing an error SQLCA at trace entry 16.

Related concepts:

- "Trace output file analysis" on page 54

Related reference:

- "Trace output file samples" on page 55

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.

Related concepts:

- "Control Center overview" in *Administration Guide: Implementation*

Related reference:

- "db2cc - Start control center command" in *Command Reference*

JDBC trace files

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.

By default, the location of the DB2 CLI/ODBC configuration keyword file 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.

Step 1. Create a path for the trace files.

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

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

On Linux and UNIX:

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

Step 2. Update the CLI configuration keywords

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

Option A: Manually editing the db2cli.ini file.

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

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

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

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

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

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

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

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

Step 3. Confirm the db2cli.ini configuration

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

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

Step 4. Restart the application

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

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

Step 5. Capture the error

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

Step 6. Disable the JDBC trace

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

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

Restart any applications that are running and tracing.

Step 7. Collect the trace files

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

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

Related concepts:

- "CLI/ODBC/JDBC trace facility" in *Call Level Interface Guide and Reference, Volume 1*
- "db2cli.ini initialization file" in *Call Level Interface Guide and Reference, Volume 1*

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.

Option A:

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

Option B:

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

For example:

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

Option C:

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

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

Related concepts:

- "Example of a trace program under the IBM DB2 Driver for JDBC and SQLJ" in *Developing Java Applications*
- "JDBC and SQLJ problem diagnosis with the IBM DB2 Driver for JDBC and SQLJ" in *Developing Java Applications*

CLI traces

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.

Related concepts:

- "CLI and JDBC trace files" in *Call Level Interface Guide and Reference, Volume 1*
- "CLI/ODBC/JDBC trace facility" in *Call Level Interface Guide and Reference, Volume 1*

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.

Step 1. Create a path for the trace files.

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

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

On Linux and UNIX:

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

Step 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
```

Step 3. Confirm the db2cli.ini configuration

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

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

Step 4. Restart the application

The db2cli.ini file is only read 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.

Step 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.

Step 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.

Step 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.

Related concepts:

- "CLI trace files" on page 64
- "CLI/ODBC/JDBC trace facility" in *Call Level Interface Guide and Reference, Volume 1*

Related reference:

- "CLI/ODBC configuration keywords listing by category" in *Call Level Interface Guide and Reference, Volume 1*

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, which can tell us in great detail 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).

Related concepts:

- "CLI and JDBC trace files" in *Call Level Interface Guide and Reference, Volume 1*
- "CLI trace files" on page 64

Related reference:

- "CLI and ODBC function summary" in *Call Level Interface Guide and Reference, Volume 2*

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
```

Related concepts:

- "CLI trace files" on page 64

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 3 display options with or without ISO timestamps and/or processor ticks.

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.

Related concepts:

- "CLI and JDBC trace files" in *Call Level Interface Guide and Reference, Volume 1*
- "CLI trace files" on page 64
- "CLI/ODBC/JDBC trace facility" in *Call Level Interface Guide and Reference, Volume 1*

Related reference:

- "TraceTimestamp CLI/ODBC configuration keyword" in *Call Level Interface Guide and Reference, Volume 1*

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.

Related concepts:

- "CLI and JDBC trace files" in *Call Level Interface Guide and Reference, Volume 1*
- "CLI trace files" on page 64
- "CLI/ODBC/JDBC trace facility" in *Call Level Interface Guide and Reference, Volume 1*

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).

Related concepts:

- "CLI and JDBC trace files" in *Call Level Interface Guide and Reference, Volume 1*
- "CLI trace files" on page 64
- "CLI/ODBC/JDBC trace facility" in *Call Level Interface Guide and Reference, Volume 1*

Platform-specific tools

Diagnostic tools (Windows)

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

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

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

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

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

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

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

Related concepts:

- "Diagnostic tools (Linux and UNIX)" on page 71
- "First failure data capture information" on page 3

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 node:

```
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 node in a single window,

and is good for high-level monitoring. However, if activity is low, the output from this monitor is of limited value.

spmon

If you are using multiple nodes on RS/6000® SP™ systems, you might need to check if the high performance switch (HPS) is running on all workstations. To view the status of all nodes, use one of the following commands from the control workstation:

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

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

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

The asterisk does not appear if the switch is up.

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 node, 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.

Related concepts:

- "First failure data capture information" on page 3
- "System core files (UNIX)" on page 16

Related reference:

- "Diagnostic tools (Windows)" on page 70

Chapter 5. Searching knowledge bases

How to search effectively for known problems

There are many resources available that describe known problems, including DB2 APARs, whitepapers, redbooks, technotes and manuals. It is important to be able to effectively search these (and other) resources in order to quickly determine whether a solution already exists for the problem you are experiencing.

Before searching, 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 quotes) you will get only those documents that contain the exact phrase in the exact order in which you type it. (As opposed to all documents that contain any combination of those three words).
4. Use wildcards. If you are encountering a specific SQL error, search for "SQL5005<wildcard>", where wildcard is the appropriate wildcard for the resource you're searching. This is likely to return more results than if you had merely searched for "SQL5005" or "SQL5005c ".
5. If you are encountering a situation where your instance ends abnormally and produces trap files, search for known problems using the first two or three functions in the trap or core file's stack traceback. If too many results are returned, try adding keywords "trap", "abend" or "crash".
6. If you are searching for keywords that are operating-system-specific (such as signal numbers or errno values), try searching on the constant name, not the value. 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

Related concepts:

- "Introduction to problem determination" on page 1

Problem determination resources

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

DB2 documentation:

Troubleshooting information can be found 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

Related concepts:

- “Overview of the DB2 technical information” on page 79
- “About the Release Notes” in *Release notes*

Chapter 6. Getting fixes

Applying fix packs

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. Every fix pack is accompanied by a document, called APARLIST.TXT, that describes the fixes it contains.

Each fix pack contains a Readme and a set of Release Notes:

- The fix pack Readme provides instructions for installing and uninstalling the fix pack.
- The Release Notes contain information about changes to the product.

You can access and read both the fix pack Readme and the Release Notes by selecting the link for fix pack downloads at the DB2 Support Web site before installing a DB2 product. The DB2 Support Web site is found at http://www.ibm.com/software/data/db2/udb/support.html.

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. It is recommended that you keep your DB2 environment running at the latest fix pack level to ensure problem-free operation.

There are two types of fix pack images:

- A fix pack for each individual DB2 product. This fix pack 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 fix pack (Linux or UNIX only). A universal fix pack services installations where more than one DB2 product has been installed.

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

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

Prerequisites:

Each fix pack has specific prerequisites. See the Readme that accompanies the fix pack for details.

Procedure:

1. Access and download the latest DB2 fix pack by selecting the link for fix pack downloads from the DB2 Support Web site at http://www.ibm.com/software/data/db2/udb/support.html.

Related reference:

- "db2setup - Install DB2 command" in *Command Reference*
- "installFixPack - Update installed DB2 products command" in *Command Reference*
- "setup - Install DB2 command" in *Command Reference*

Appendix A. DB2 Database technical information

Overview of the DB2 technical information

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

- DB2 Information Center
  - Topics
  - Help for DB2 tools
  - Sample programs
  - Tutorials
- DB2 books
  - PDF files (downloadable)
  - PDF files (from the DB2 PDF CD)
  - printed books
- Command line help
  - Command help
  - Message help
- Sample programs

IBM periodically makes documentation updates available. If you access the online version on the DB2 Information Center at ibm.com®, you do not need to install documentation updates because this version is kept up-to-date by IBM. If you have installed the DB2 Information Center, it is recommended that you install the documentation updates. Documentation updates allow you to update the information that you installed from the *DB2 Information Center CD* or downloaded from Passport Advantage as new information becomes available.

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

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

Documentation feedback

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

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

Related concepts:

- "Features of the DB2 Information Center" in *Online DB2 Information Center*
- "Sample files" in *Samples Topics*

Related tasks:

- "Invoking command help from the command line processor" in *Command Reference*
- "Invoking message help from the command line processor" in *Command Reference*
- "Updating the DB2 Information Center installed on your computer or intranet server" on page 85

Related reference:

- "DB2 technical library in hardcopy or PDF format" on page 80

DB2 technical library in hardcopy or PDF format

The following tables describe the DB2 library available from the IBM Publications Center at www.ibm.com/shop/publications/order. DB2 Version 9 manuals in PDF format can be downloaded from www.ibm.com/software/data/db2/udb/support/manualsv9.html.

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

The information in these books is fundamental to all DB2 users; you will find this information useful whether you are a programmer, a database administrator, or someone who works with DB2 Connect or other DB2 products.

Table 1. DB2 technical information

Name	Form Number	Available in print
Administration Guide: Implementation	SC10-4221	Yes
Administration Guide: Planning	SC10-4223	Yes
Administrative API Reference	SC10-4231	Yes
Administrative SQL Routines and Views	SC10-4293	No
Call Level Interface Guide and Reference, Volume 1	SC10-4224	Yes
Call Level Interface Guide and Reference, Volume 2	SC10-4225	Yes
Command Reference	SC10-4226	No
Data Movement Utilities Guide and Reference	SC10-4227	Yes
Data Recovery and High Availability Guide and Reference	SC10-4228	Yes
Developing ADO.NET and OLE DB Applications	SC10-4230	Yes
Developing Embedded SQL Applications	SC10-4232	Yes

Table 1. DB2 technical information (continued)

Name	Form Number	Available in print
Developing SQL and External Routines	SC10-4373	No
Developing Java Applications	SC10-4233	Yes
Developing Perl and PHP Applications	SC10-4234	No
Getting Started with Database Application Development	SC10-4252	Yes
Getting started with DB2 installation and administration on Linux and Windows	GC10-4247	Yes
Message Reference Volume 1	SC10-4238	No
Message Reference Volume 2	SC10-4239	No
Migration Guide	GC10-4237	Yes
Net Search Extender Administration and User's Guide **Note:** HTML for this document is not installed from the HTML documentation CD.	SH12-6842	Yes
Performance Guide	SC10-4222	Yes
Query Patroller Administration and User's Guide	GC10-4241	Yes
Quick Beginnings for DB2 Clients	GC10-4242	No
Quick Beginnings for DB2 Servers	GC10-4246	Yes
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC18-9749	Yes
SQL Guide	SC10-4248	Yes
SQL Reference, Volume 1	SC10-4249	Yes
SQL Reference, Volume 2	SC10-4250	Yes
System Monitor Guide and Reference	SC10-4251	Yes
Troubleshooting Guide	GC10-4240	No
Visual Explain Tutorial	SC10-4319	No
What's New	SC10-4253	Yes
XML Extender Administration and Programming	SC18-9750	Yes
XML Guide	SC10-4254	Yes
XQuery Reference	SC18-9796	Yes

Table 2. DB2 Connect-specific technical information

Name	Form Number	Available in print
DB2 Connect User's Guide	SC10-4229	Yes

Table 2. DB2 Connect-specific technical information (continued)

Name	Form Number	Available in print
Quick Beginnings for DB2 Connect Personal Edition	GC10-4244	Yes
Quick Beginnings for DB2 Connect Servers	GC10-4243	Yes

Table 3. WebSphere® Information Integration technical information

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

Note: The DB2 Release Notes provide additional information specific to your product's release and fix pack level. For more information, see the related links.

Related concepts:

- "Overview of the DB2 technical information" on page 79
- "About the Release Notes" in *Release notes*

Related tasks:

- "Ordering printed DB2 books" on page 82

Ordering printed DB2 books

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

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

Note: The most up-to-date and complete DB2 documentation is maintained in the DB2 Information Center at http://publib.boulder.ibm.com/infocenter/db2help/.

Procedure:

To order printed DB2 books:

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

Related concepts:

- "Overview of the DB2 technical information" on page 79

Related reference:

- "DB2 technical library in hardcopy or PDF format" on page 80

Displaying SQL state help from the command line processor

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

Procedure:

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

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

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

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

Related tasks:

- "Invoking command help from the command line processor" in *Command Reference*
- "Invoking message help from the command line processor" in *Command Reference*

Accessing different versions of the DB2 Information Center

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

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

Related tasks:

- "Setting up access to DB2 contextual help and documentation" in *Administration Guide: Implementation*

Displaying topics in your preferred language in the DB2 Information Center

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

Procedure:

To display topics in your preferred language in the Internet Explorer browser:

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

     Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.
   - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

To display topics in your preferred language in a Firefox or Mozilla browser:

1. Select the **Tools** —> **Options** —> **Languages** button. The Languages panel is displayed in the Preferences window.
2. Ensure your preferred language is specified as the first entry in the list of languages.
   - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
   - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

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

Related concepts:

- "Overview of the DB2 technical information" on page 79

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

If you have a locally-installed DB2 Information Center, updated topics can be available for download. The 'Last updated' value found at the bottom of most topics indicates the current level for that topic.

To determine if there is an update available for the entire DB2 Information Center, look for the 'Last updated' value on the Information Center home page. Compare the value in your locally installed home page to the date of the most recent downloadable update at http://www.ibm.com/software/data/db2/udb/support/icupdate.html. You can then update your locally-installed Information Center if a more recent downloadable update is available.

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

1. Stop the DB2 Information Center on your computer, and restart the Information Center in stand-alone mode. Running the Information Center in stand-alone mode prevents other users on your network from accessing the Information Center, and allows you to download and apply updates.
2. Use the Update feature to determine if update packages are available from IBM.

   Note: Updates are also available on CD. For details on how to configure your Information Center to install updates from CD, see the related links.

   If update packages are available, use the Update feature to download the packages. (The Update feature is only available in stand-alone mode.)
3. Stop the stand-alone Information Center, and restart the DB2 Information Center service on your computer.

Procedure:

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

1. Stop the DB2 Information Center service.
   - On Windows, click **Start → Control Panel → Administrative Tools → Services**. Then right-click on **DB2 Information Center** service and select **Stop**.
   - On Linux, enter the following command:

     ```
     /etc/init.d/db2icdv9 stop
     ```
2. Start the Information Center in stand-alone mode.
   - On Windows:
     a. Open a command window.
     b. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the C:\Program Files\IBM\DB2 Information Center\Version 9 directory.
     c. Run the help_start.bat file using the fully qualified path for the DB2 Information Center:

        ```
        <DB2 Information Center dir>\doc\bin\help_start.bat
        ```
   - On Linux:

a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the /opt/ibm/db2ic/V9 directory.
b. Run the help_start script using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>/doc/bin/help_start
```

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

3. Click the Update button (). On the right hand panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the download process, check the selections you want to download, then click **Install Updates**.
5. After the download and installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center.
   - On Windows, run the help_end.bat file using the fully qualified path for the DB2 Information Center:

     ```
     <DB2 Information Center dir>\doc\bin\help_end.bat
     ```

     Note: The help_end batch file contains the commands required to safely terminate the processes that were started with the help_start batch file. Do not use `Ctrl-C` or any other method to terminate help_start.bat.
   - On Linux, run the help_end script using the fully qualified path for the DB2 Information Center:

     ```
     <DB2 Information Center dir>/doc/bin/help_end
     ```

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

     ```
     /etc/init.d/db2icdv9 start
     ```

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

Related concepts:

- "DB2 Information Center installation options" in *Quick Beginnings for DB2 Servers*

Related tasks:

- "Installing the DB2 Information Center using the DB2 Setup wizard (Linux)" in *Quick Beginnings for DB2 Servers*
- "Installing the DB2 Information Center using the DB2 Setup wizard (Windows)" in *Quick Beginnings for DB2 Servers*

DB2 tutorials

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

Before you begin:

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

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

DB2 tutorials:

To view the tutorial, click on the title.

Native XML data store
Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

Visual Explain Tutorial
Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

Related concepts:

- "Visual Explain overview" in *Administration Guide: Implementation*

DB2 troubleshooting information

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

DB2 documentation
Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

DB2 Technical Support Web site
Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs or bug fixes), fix packs, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at http://www.ibm.com/software/data/db2/udb/support.html

Related concepts:

- "Introduction to problem determination" on page 1
- "Overview of the DB2 technical information" on page 79

Terms and Conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal use: You may reproduce these Publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these Publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Appendix B. Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _*enter the year or years*_. All rights reserved.

Trademarks

Company, product, or service names identified in the documents of the DB2 Version 9 documentation library may be trademarks or service marks of International Business Machines Corporation or other companies. Information on the trademarks of IBM Corporation in the United States, other countries, or both is located at http://www.ibm.com/legal/copytrade.shtml.

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 documentation library:

Microsoft®, Windows, Windows NT®, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel®, Itanium®, Pentium®, and Xeon® are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Index

Contacting IBM

To contact IBM in your country or region, check the IBM Directory of Worldwide Contacts at http://www.ibm.com/planetwide

To learn more about DB2 products, go to http://www.ibm.com/software/data/db2/.

Printed in USA

GC10-4240-00

Spine information:

IBM DB2 **DB2 Version 9** **Troubleshooting Guide**