IBM

Troubleshooting and support

Version 11 Release 1

Contents

Troubleshooting and support

To help you understand, isolate, and resolve problems with your Db2® software, the troubleshooting and support section contains instructions for using the problem-determination resources that are provided.

To resolve a problem on your own, you need to identify the source, gather diagnostic information, get fixes, and search the appropriate knowledge bases. If you must contact IBM Software Support, you can find out what diagnostic information the service technicians require to help you address your problem.

- Searching knowledge bases

  You can often find solutions to problems by searching IBM knowledge bases. This topic describes how to optimize your results by using available resources, support tools, and search methods.
- Troubleshooting a problem

  Troubleshooting is a systematic approach to solving a problem. The goal is to determine why something does not work as expected and how to resolve the problem.
- Learning more about troubleshooting tools

  The following topics can help you acquire the conceptual information that you require to effectively troubleshoot problems with Db2 database server.
- Getting a fix

  A product fix might be available to resolve your problem. You can get fixes by following these steps.
- Contacting IBM Software Support

  IBM Software Support provides assistance with product defects.

How to search effectively for known problems

There are many resources available that describe known problems, including Db2 APARs, whitepapers, IBM® Redbooks® publications, Technotes, and manuals. It is important to be able to effectively search these (and other) resources in order to quickly determine whether a solution already exists for the problem you are experiencing.

Before searching, you should have a clear understanding of your problem situation.

Once you have a clear understanding of what the problem situation is, you need to create a list of search keywords to increase your chances of finding the existing solutions. Here are some tips:

1. Use multiple words in your search. The more pertinent search terms you use, the better your search results will be.
2. Start with specific results, and then go to broader results if necessary. For example, if too few results are returned, then remove some of the less pertinent search terms and try it again. Alternatively, if you are uncertain which keywords to use, you can perform a broad search with a few keywords, look at the type of results that you receive, and be able to make a more informed choice of additional keywords.

3. Sometimes it is more effective to search for a specific phrase. For example, if you enter: "administration notification file" (with the quotation marks) you will get only those documents that contain the exact phrase in the exact order in which you type it. (As opposed to all documents that contain any combination of those three words).
4. Use wildcards. If you are encountering a specific SQL error, search for "`SQL5005<wildcard>`", where `<wildcard>` is the appropriate wildcard for the resource you're searching. This is likely to return more results than if you had merely searched for "`SQL5005`" or "`SQL5005c` ".
5. If you are encountering a situation where your instance ends abnormally and produces trap files, search for known problems using the first two or three functions in the trap or core file's stack traceback. If too many results are returned, try adding keywords "`trap`", "`abend`" or "`crash`".
6. If you are searching for keywords that are operating-system-specific (such as signal numbers or errno values), try searching on the constant name, not the value. For example, search for "EFBIG" instead of the error number 27.

In general, search terms that are successful often involve:
- Words that describe the command run
- Words that describe the symptoms
- Tokens from the diagnostics

Available troubleshooting resources

A wide variety of troubleshooting information is available to assist you in using Db2 database products.

Db2 documentation

Troubleshooting information can be found throughout the *Db2 Information Center*, as well as throughout the PDF books that make up the Db2 library.

Db2 Technical Support website

Refer to the Db2 Technical Support website 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 other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the Db2 Technical Support website at: www.ibm.com/software/data/db2/support/db2_9/

Troubleshooting techniques

The first step in good problem analysis is to describe the problem completely. Without a problem description, you will not know where to start investigating the cause of the problem.

This step includes asking yourself such basic questions as:
- What are the symptoms?
- Where is the problem happening?
- When does the problem happen?
- Under which conditions does the problem happen?

- Is the problem reproducible?

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

What are the symptoms?

When starting to describe a problem, the most obvious question is "What is the problem?" This might seem like a straightforward question; however, it can be broken down into several other questions to create a more descriptive picture of the problem. These questions can include:
- Who or what is reporting the problem?
- What are the error codes and error messages?
- How does it fail? For example: loop, hang, stop, performance degradation, incorrect result.
- What is the affect on business?

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?
- Is the database stored on individual disks, or on a RAID disk array?

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

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

When does the problem happen?

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

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

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

Under which conditions does the problem happen?

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

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

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

Is the problem reproducible?

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

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

- Can the problem be recreated on a test machine?
- Are multiple users or applications encountering the same type of problem?
- Can the problem be recreated by running a single command, a set of commands, or a particular existing application or deliberately crafted test application?
- Can the problem be recreated by executing the equivalent command/query with the Db2 command line processor?

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 information:

Best practices: Troubleshooting Db2 servers

Troubleshooting Db2 Net Search Extender

Example

Warnings and errors when performing Net Search Extender post-installation verification

The **nsesample** post-installation verification script (part of the Net Search Extender installation) might generate warnings and errors such as SQL3149N and SQL0803N. The Net Search Extender Administration and User's Guide states that the script should run without any errors.

Symptoms

The **nsesample** script reports warnings and errors if you have an existing connection to a database at the time you run the script. For example:

- SQL3148W: A row from the input file was not inserted into the table. SQLCODE *sqlcode* was returned.
- SQL0803N: One or more values in the **INSERT** statement, **UPDATE** statement, or foreign key update that is caused by a **DELETE** statement are not valid because the primary key, unique constraint, or unique index that is identified by *index-id* constrains table *table-name* from having duplicate rows for those columns.

If you have an existing database connection, the script fails to create the new database.

Causes

The `nsesample.bat` script does no error checking, and does not include an explicit **CONNECT RESET** command prior to attempting to create the database that was specified when running the validation script `nsesample.bat`. Subsequent steps fail as it tries to connect to the database and insert data. The script does not perform any clean-up either, which might be necessary to facilitate the subsequent steps in the PDF instructions. Clean-up is needed before any retesting is attempted.

Resolving the problem

To resolve this issue, remove or rename the `nsesample.log` and start a new CLP session. Issue the command **DB2 CONNECT RESET** to ensure that there are no existing database connections and try the script again. If the script was run previously, use a new database name, or drop the tables that are created as part of this testing (namely db2ext.texttab and db2ext.htmltab).

Failure during fix pack installation of Net Search Extender

Symptoms

Install fails and the following error message is returned.

```
Error while trying to backup file "/opt/ibm/db2/V9.x/./adm/ctecsdem".

Db2 Net Search Extender Version 9.7.0.3 installation to /opt/ibm/db2/V9.x failed.

For details see /tmp/db2nseXXXXXXXXXXXXXXX.log.
```

The installation log contains the following content:

```
/opt/ibm/db2/V9.7/./adm/ctecsdem to /tmp/nsetempzmeDwj/./adm/ctecsdem
    failed. Invalid cross-device link
```

Causes

An older version of Net Search Extender (NSE) exists. NSE fix packs are complete product installs, which cannot upgrade an older version, but replace an older version.

Resolving the problem

To install an NSE fix pack, uninstall the current version of NSE before the installing the fix pack. From command line, for each of the Db2 instance where NSE needs to be uninstalled and upgraded, run the following commands or tasks.

1. Switch to the user ID of the Db2 instance
2. db2text stop
3. db2stop
4. Ensure that you are active as the root user
5. Change your working directory to the Db2 path where you want to remove Net Search Extender. For example, cd /opt/IBM/db2/V9.1/install
6. Issue the command ./db2nse_deinstall
7. Run the NSE Install Command
8. db2iupdt <db2 instance name>
9. Log on as db2 instance owner
10. db2start
11. db2text start

Troubleshooting tools

Tools are available to help collect, format or analyze diagnostic data.

Overview of troubleshooting commands

The Db2 software provides multiple commands that you can use for troubleshooting your database.

The following command table provides various commands that you can aid you in troubleshooting your Db2database.

Table 1. Db2 troubleshooting commands

Command	What the command does	When to use it	Administrative authority that is required to run the command
db2cklog	This command is used for checking archive log files to ensure that they are valid for rollforward recovery.	You can use the command periodically to ensure that archive log files are valid for use in a rollforward recovery if there ever were a situation where a rollforward is required.	Users with read permissions on the archive log files can run this command.
db2dart	This command is used to verify the architectural correctness of databases and report any errors.	You can be use this command to inspect the entire database, a table space, or a table for correctness.	Users with SYSADM authority can run this command.
db2diag	This command filters and formats the diagnostic information available in the `db2diag.log` files. It can be used to filter and format both single and rotating log files.	You can be use this command to filter `db2diag.log` files. This reduces the time that is required to locate the records that are needed when used by IBM Software Support for troubleshooting.	Any user can run this command.

Table 1. Db2 troubleshooting commands (continued)

Command	What the command does	When to use it	Administrative authority that is required to run the command
db2fodc	This command is used for manual first occurrence data collection (FODC) on problems that cannot trigger automatic FODC.	You can be use this command collect information about potential hangs, severe performance issues, and various types of errors.	Instance owners with SYSADM authority (on UNIX and Linux operating systems) can run this command. On Windows operating systems, users with SYSADM authority can run this command.
db2greg	This command is used to view the global registry and is only available on UNIX and Linux operating systems.	You can use this command to locate where data is being indexed.	Any user can run this command.
db2level	This command is used to show the version of Db2 software that is running and the service level of that instance.	You can use this command for troubleshooting your Db2 instance if the IBM Software Support requires knowledge of your version and instance.	Any user can run this command.
db2look	This command is used to create a mirror of the current database image.	You can use this command for testing out new software and compatibilities so there is no harming or corrupting the data within the real database.	Any user with SYSADM, SYSCTRL, SYSMAINT, SYSMON, DBADM, or EXECUTE privilege on the ADMIN_GET_STORAGE_PATHS table function can run this command.
db2ls	This command is used to list the products and features that are installed on your system and is only available on UNIX and Linux operating systems.	You can use this command to list the products and features that are installed on your system to review possible items which might require troubleshooting.	Any user can run this command.
db2pd	This command is used to monitor and troubleshoot a Db2 instance from memory.	You can use this command to review processes and information while it is changing. Since this tool is not dependent on the Db2 engine to be up, it can be used on an engine that is hung and collect information without acquiring any latches or using any engine resources.	Any user with SYSADM, SYSCTRL, SYSMAINT, or SYSMON can run this command.

Table 1. Db2 troubleshooting commands (continued)

Command	What the command does	When to use it	Administrative authority that is required to run the command
db2support	This command is used to automatically collect all Db2 and system diagnostic information available.	You can use this command to collect diagnostic data that is uploaded for IBM Support Staff to analyze the diagnostic data locally.	For most thorough results, run this command as SYSADM. Any user can run this command, but results might vary based on privileges.
db2trc	This command is used to record information about operations and formats the information into a readable form.	You can run this command to record information about the system for troubleshooting. Run this command under the guidance of support personnel.	Any user with SYSADM, SYSCTRL, SYSMAINT, or DASADM (on UNIX and Linux operating systems) can run this command.
db2val	This command is used to ensure that your copy of the Db2 software is functioning properly.	You can run this command to validate installation files, instances, database creation, connections to the database, and the state of partitioned database environments.	Any user with root authority (on UNIX and Linux operating systems), or SYSADM plus either instance owner or root/local administrator access.

Learning more about troubleshooting tools

The following topics can help you to acquire the conceptual information that you need to effectively troubleshoot problems with the Db2 product:

- About troubleshooting

  Troubleshooting is a systematic approach to solving a problem. The goal is to determine why something does not work as expected and how to resolve the problem.
- About the diagnostic data directory path

  Depending on your platform, Db2 diagnostic information contained in a dump file, trap file, diagnostic log file, administration notification log file, alert log file, and first occurrence data collection (FODC) package can be found in the diagnostic data directory specified by the **diagpath** database manager configuration parameter.
- About administration notification log files

  The Db2 database manager writes the following kinds of information to the administration notification log: the status of Db2 utilities such as **REORG** and **BACKUP**; client application errors; service class changes, licensing activity; log file paths and storage problems; monitoring and indexing activities; and table space problems. A database administrator can use this information to diagnose problems, tune the database, or monitor the database.
- About DB2® diagnostic (db2diag) log files

  With the addition of administration notification log messages being logged to the **db2diag** log files using a standardized message format, viewing the **db2diag** log files is an excellent first task in understanding what has been happening to the database.

- About platform-specific error logs

  There are many other files and utilities available outside of Db2 to help analyze problems. Often they are just as important to determining root cause as the information made available in the Db2 files.
- About messages

  Learning more about messages can help you to identify an error or problem and resolve the problem by using the appropriate recovery action. This information can also be used to understand where messages are generated and logged.
- About internal return codes

  There are two types of internal return codes: ZRC values and ECF values. They are displayed in Db2 trace output and in the **db2diag** log files. ZRC and ECF values are typically negative numbers and are used to represent error conditions.
- About 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.
- About trap files

  Db2 generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception. All signals or exceptions received by Db2 are recorded in the trap file. The trap file also contains the function sequence that was running when the error occurred. This sequence is sometimes referred to as the "function call stack" or "stack trace." The trap file also contains additional information about the state of the process when the signal or exception was caught.
- About first occurrence data capture (FODC)

  First occurrence data capture (FODC) is the process used to capture scenario-based data about a Db2 instance. FODC can be invoked manually by a Db2 user based on a particular symptom or invoked automatically when a predetermined scenario or symptom is detected. This information reduces the need to reproduce errors to get diagnostic information.
- About callout script (db2cos) output files

  A db2cos script is invoked by default when the database manager cannot continue processing due to a panic, trap, segmentation violation or exception.
- About combining DB2 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.

Diagnostic data directory path:

Depending on your platform, Db2 diagnostic information contained in a dump file, trap file, diagnostic log file, administration notification log file, alert log file, and first occurrence data collection (FODC) package can be found in the diagnostic data directory specified by the **diagpath** or **alt_diagpath** database manager configuration parameters.

Overview

The specification of the diagnostic data directory path or the alternate diagnostic data directory path, using the **diagpath** or **alt_diagpath** database manager configuration parameters, can determine which one of the following directory path methods can be used for diagnostic data storage:

Primary diagnostic data directory path
All diagnostic data for members, cluster caching facilities, database partition servers, and database partitions is logged to a private db2diag log file. This split diagnostic data directory path is the default condition unless you specify the **diagpath** value with a valid path name and the `$h`, `$n`, or `$m` pattern identifiers.

Alternate diagnostic data directory path
The **alt_diagpath** database manager configuration parameter is an alternate diagnostic data directory path that provides a secondary path for storing diagnostic information. The path specified by the **alt_diagpath** parameter is used only when the database manager fails to write to the path specified in **diagpath** and ensures that important diagnostic information is not lost. For the alternate diagnostic data directory path to be available, you must set the **alt_diagpath** configuration parameter. For greater resiliency, it is recommended that you set this parameter to a path that is on a different file system than **diagpath**.

Benefits

The benefit of specifying a single diagnostic data directory path is that diagnostic information, from several database partitions and hosts, can be consolidated in a central location for easy access by setting a single diagnostic data directory path. The benefit of using the default split diagnostic data directory path is that diagnostic logging performance can be improved because of less contentions on the **db2diag** log file.

The benefits of specifying a secondary diagnostic data path, **alt_diagpath**, are:
- Increased resiliency to the loss of important diagnostic information.
- Compatibility with some tools used for **diagpath** such as splitting.

Merging files and sorting records

Merging and sorting records of multiple diagnostic files of the same type, based on timestamps, can be done with the **db2diag -merge** command in the case of a split diagnostic data directory path. For more information, see: "db2diag - db2diag logs analysis tool command" in the *Command Reference*.

Space requirements for diagnostic data

Diagnostic data collection in the path specified by the **diagpath** parameter can generate large volumes of diagnostic information, especially if core file dumps and first occurrence data capture (FODC) data is not redirected to a separate directory path or if you use a single `db2diag.log` file that grows in size indefinitely. Enough space must be available to store the diagnostic data, and you must perform regular house keeping in the diagnostic path to ensure sufficient space remains available.

You can use the following recommendations when configuring diagnostic data logging on your data server to make sure space requirements for diagnostic data are met:

Meet minimum space requirements for diagnostic
: The minimum amount of free space available in the diagnostic directory path should be two times the amount of physical memory installed on the machine (minimum free space = 2x physical memory). For example, if a machine has 64 GB of physical memory installed, a minimum of 128 GB of space for diagnostic data should be available in the file system.

Redirect core file dumps and FODC data to a different directory path
: Both core file dumps and FODC data can consume significant disk space quickly and both send data to the directory path specified by the **diagpath** database manager configuration parameter by default. To keep more space available in the diagnostic directory path, core file dumps and FODC data can be redirected to a different directory path or file system. You can control where core files are generated through the *DB2FODC* registry variable by setting the *DUMPDIR* variable to point to a directory path that is different from **diagpath**. Similarly, you can control where FODC package directories are created by setting *FODCPATH* variable to point to a different directory path.

Move or remove files that are no longer needed
: If you run the db2support command without specifying an output path that is different than **diagpath**, the resulting compressed archive is stored in the diagnostic directory path. Once you have uploaded the file to IBM, remember to move the compressed archive out of the diagnostic directory path or it will continue to consume available disk space.

Configure for rotating diagnostic logs and archive log files
: By default, if you use a single `db2diag.log` file, the Db2 diagnostic log file will grow in size indefinitely. If you configure for rotating diagnostic logs by setting the **diagsize** database manager configuration parameter, then a series of rotating diagnostic log files and a series of rotating administration notification log files are used that fit into the size defined by **diagsize**. As log files fill up, the oldest files are deleted and new log files are created. To avoid losing information too quickly because of file rotation (the deletion of the oldest log file), set **diagsize** to a value that is greater than 50 MB but not more than 80% of the free space in the directory paths that you specify with the **diagpath** and **alt_diagpath** parameters. You can also preserve rotating diagnostic log files by archiving them from **diagpath** with the **db2diag -archive** command.

Configure an alternate diagnostic path
: As a fail-safe to prevent the loss of important diagnostic information, the **alt_diagpath** database manager configuration parameter provides an alternate diagnostic data directory path for storing diagnostic information. If the database manager fails to write to the path specified by **diagpath**, the path specified by **alt_diagpath** is used to store diagnostic information until **diagpath** becomes available again. For greater resiliency, point the **alt_diagpath** parameter to a different file system than the **diagpath** parameter.

Splitting a diagnostic data directory path by database partition server, database partition, or both:

You can specify a diagnostic data directory path so that separate directories are created and named according to the database partition server, database partition, or both.

Before you begin

Db2 Version 9.7 Fix Pack 1 or a later fix pack is required.

About this task

You can specify a diagnostic data directory path to separately store diagnostic information according to the database partition server or database partition from which the diagnostic data dump originated.

Procedure

- **Splitting diagnostic data directory path per physical database partition server**
  - To specify a default diagnostic data directory path, execute the following step:
    - Set the **diagpath** database manager configuration parameter to split the default diagnostic data directory path per physical database partition server by issuing the following command:
      ```
      db2 update dbm cfg using diagpath '"$h"'
      ```

    This command creates a subdirectory under the default diagnostic data directory with the computer name, as shown in the following example:

    `*Default_diagpath*/HOST_*db-partition-server-name*`
  - To split a user specified diagnostic data directory path (for example, `/home/usr1/db2dump/`), execute the following step:
    - Set the **diagpath** database manager configuration parameter to split the `/home/usr1/db2dump/` diagnostic data directory path per database partition server by issuing the following command:
      ```
      db2 update dbm cfg using diagpath '"/home/usr1/db2dump/ $h"'
      ```

      Note: A blank space must separate `/home/usr1/db2dump/` and `$h`.

    This command creates a subdirectory under the `/home/usr1/db2dump/` diagnostic data directory with the database partition server name, as shown in the following example:

    `/home/usr1/db2dump/HOST_*db-partition-server-name*`
- **Splitting diagnostic data directory path per database partition**
  - To specify a default diagnostic data directory path, execute the following step:
    - Set the **diagpath** database manager configuration parameter to split the default diagnostic data directory path per database partition by issuing the following command:
      ```
      db2 update dbm cfg using diagpath '"$n"'
      ```

    This command creates a subdirectory for each partition under the default diagnostic data directory with the partition number, as shown in the following example:

    `*Default_diagpath*/NODE*number*`
  - To split a user specified diagnostic data directory path (for example, `/home/usr1/db2dump/`), execute the following step:

- Set the **diagpath** database manager configuration parameter to split the /home/usr1/db2dump/ diagnostic data directory path per database partition by issuing the following command:

  ```
  db2 update dbm cfg using diagpath '"/home/usr1/db2dump/ $n"'
  ```

  Note: A blank space must separate /home/usr1/db2dump/ and $n.

This command creates a subdirectory for each partition under the /home/usr1/db2dump/ diagnostic data directory with the partition number, as shown in the following example:

/home/usr1/db2dump/NODE*number*

- **Splitting diagnostic data directory path per physical database partition server and per database partition**
  – To specify a default diagnostic data directory path, execute the following step:
    - Set the **diagpath** database manager configuration parameter to split the default diagnostic data directory path per physical database partition server and per database partition by issuing the following command:

      ```
      db2 update dbm cfg using diagpath '"$h$n"'
      ```

    This command creates a subdirectory for each logical partition on the database partition server under the default diagnostic data directory with the database partition server name and partition number, as shown in the following example:

    Default_diagpath/HOST_*db-partition-server-name*/NODE*number*
  – To specify a user specified diagnostic data directory path (for example, /home/usr1/db2dump/), execute the following step:
    - Set the **diagpath** database manager configuration parameter to split the /home/usr1/db2dump/ diagnostic data directory path per database partition server and per database partition by issuing the following command:

      ```
      db2 update dbm cfg using diagpath '"/home/usr1/db2dump/ $h$n"'
      ```

      Note: A blank space must separate /home/usr1/db2dump/ and hn.

    This command creates a subdirectory for each logical partition on the database partition server under the /home/usr1/db2dump/ diagnostic data directory with the database partition server name and partition number, as shown in the following example:

    /home/usr1/db2dump/HOST_*db-partition-server-name*/NODE*number*

    For example, an AIX® database partition server, named boson, has 3 database partitions with node numbers 0, 1, and 2. The following example shows a sample list output for the directory:

```
usr1@boson /home/user1/db2dump->ls -R *
HOST_boson:

HOST_boson:
NODE0000 NODE0001 NODE0002

HOST_boson/NODE0000:
db2diag.log db2eventlog.000 db2resync.log db2sampl_Import.msg events usr1.nfy

HOST_boson/NODE0000/events:
db2optstats.0.log

HOST_boson/NODE0001:
db2diag.log db2eventlog.001 db2resync.log usr1.nfy stmmlog

HOST_boson/NODE0001/stmmlog:
```

```
stmm.0.log

HOST_boson/NODE0002:
db2diag.log db2eventlog.002 db2resync.log usr1.nfy
```

What to do next

Note:
- If a diagnostic data directory path split per database partition is specified (`$n` or `$h$n`), the `NODE0000` directory will always be created for each database partition server. The `NODE0000` directory can be ignored if database partition 0 does not exist on the database partition server where the `NODE0000` directory was created.
- To check that the setting of the diagnostic data directory path was successfully split, execute the following command:

  ```
  db2 get dbm cfg | grep DIAGPATH
  ```

  A successfully split diagnostic data directory path returns the values `$h`, `$n`, or `$h$n` with a preceding blank space. The following example shows a sample output that is returned:

  ```
  Diagnostic data directory path          (DIAGPATH) = /home/usr1/db2dump/ $h$n
  ```

To merge separate **db2diag** log files to make analysis and troubleshooting easier, use the **db2diag -merge** command. For additional information, see: "db2diag - db2diag logs analysis tool command" in the *Command Reference* and "Analyzing db2diag log files using db2diag tool" on page 55..

Interpreting administration notification log file entries:

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

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

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

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

```
2006-02-15-19.33.37.630000 1    Instance:DB2 2   Node:000 3
PID:940(db2syscs.exe) TID: 660 4   Appid:*LOCAL.DB2.020205091435 5
recovery manager 6  sqlpresr 7  Probe:1 8   Database:SAMPLE  9
ADM1530E 10  Crash recovery has been initiated. 11
```

Legend:
1. A timestamp for the message.
2. The name of the instance generating the message.
3. For multi-partition systems, the database partition generating the message. (In a nonpartitioned database, the value is "000".)

4. The process identifier (PID), followed by the name of the process, followed by the thread identifier (TID) that are responsible for the generation of the message.
5. Identification of the application for which the process is working. In this example, the process generating the message is working on behalf of an application with the ID *LOCAL.DB2.020205091435.

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

   To identify more about a particular application ID, either:
   - Use the **LIST APPLICATIONS** command on a Db2 server or **LIST DCS APPLICATIONS** on a Db2 Connect gateway to view a list of application IDs. From this list, you can determine information about the client experiencing the error, such as its node name and its TCP/IP address.
   - Use the **GET SNAPSHOT FOR APPLICATION** command to view a list of application IDs.
6. The Db2 component that is writing the message. For messages written by user applications using the db2AdminMsgWrite API, the component will read "User Application".
7. The name of the function that is providing the message. This function operates within the Db2 component that is writing the message. For messages written by user applications using the db2AdminMsgWrite API, the function will read "User Function".
8. Unique internal identifier. This number allows Db2 customer support and development to locate the point in the Db2 source code that reported the message.
9. The database on which the error occurred.
10. When available, a message indicating the error type and number as a hexadecimal code.
11. When available, message text explaining the logged event.

Setting the error capture level for the administration notification log file:

This task describes how to set the error capture level for the administration notification log file.

About this task

The information that Db2 records in the administration notification log is determined by the **notifylevel** setting.

Procedure

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

  Look for the following variable:
  ```
  Notify Level          (NOTIFYLEVEL) = 3
  ```
- To alter the setting, use the **UPDATE DBM CFG** command. For example:
  ```
  DB2 UPDATE DBM CFG USING NOTIFYLEVEL X
  ```
  where *X* is the notification level you want.

Db2 diagnostic (db2diag) log files:

The Db2 diagnostic **db2diag** log files are primarily intended for use by IBM Software Support for troubleshooting purposes. The administration notification log is primarily intended for troubleshooting use by database and system administrators. Administration notification log messages are also logged to the **db2diag** log files using a standardized message format.

Overview

With Db2 diagnostic and administration notification messages both logged within the **db2diag** log files, this often makes the **db2diag** log files the first location to examine in order to obtain information about the operation of your databases. Help with the interpretation of the contents of these diagnostic log files is provided in the topics listed in the "Related links" section. If your troubleshooting attempts are unable to resolve your problem and you feel you require assistance, you can contact IBM Software Support (for details, see the "Contacting IBM Software Support" topic). In gathering relevant diagnostic information that will be requested to be sent to IBM Software Support, you can expect to include your **db2diag** log files among other sources of information which includes other relevant logs, storage dumps, and traces.

The **db2diag** log file can exist in two different forms:

Single diagnostic log file
: One active diagnostic log file, named db2diag.log, that grows in size indefinitely. This is the default form and it exists whenever the **diagsize** database manager configuration parameter has the value of 0 (the default value for this parameter is 0).

Rotating diagnostic log files
: A single active log file (named db2diag.*N*.log, where *N* is the file name index that is a continuously growing number starting from 0), although a series of diagnostic log files can be found in the location defined by the **diagpath** configuration parameter, each growing until reaching a limited size, at which time the log file is closed and a new one is created and opened for logging with an incremented file name index (db2diag.*N+1*.log). It exists whenever the **diagsize** database manager configuration parameter has a nonzero value.

You can choose which of these two forms exist on your system by appropriately setting the **diagsize** database manager configuration parameter.

Configuration

The **db2diag** log files can be configured in size, location, and the types of diagnostic errors recorded by setting the following database manager configuration parameters:

diagsize
: The value of **diagsize** decides what form of diagnostic log file will be adopted. If the value is 0, a single diagnostic log file will be adopted. If the value is not 0, rotating diagnostic log files will be adopted, and this nonzero value also specifies the total size of all rotating diagnostic log files and all rotating administration notification log files. The instance must be

restarted for the new value of the **diagsize** parameter to take effect. See the "diagsize - Diagnostic log file size configuration parameter" topic for complete details.

diagpath
: Diagnostic information can be specified to be written to **db2diag** log files in the location defined by the **diagpath** configuration parameter. See the "diagpath - Diagnostic data directory path configuration parameter" topic for complete details.

alt_diagpath
: The **alt_diagpath** database manager configuration parameter provides an alternate diagnostic data directory path for storing diagnostic information. If the database manager fails to write to the path specified by **diagpath**, the path specified by **alt_diagpath** is used to store diagnostic information.

diaglevel
: The types of diagnostic errors written to the **db2diag** log files can be specified with the **diaglevel** configuration parameter. See the "diaglevel - Diagnostic error capture level configuration parameter" topic for complete details.

Note: If the **diagsize** configuration parameter is set to a non-zero value, that value specifies the total size of the combination of all rotating administration notification log files and all rotating diagnostic log files contained within the diagnostic data directory. For example, if a system with 4 database partitions has **diagsize** set to 1 GB, the maximum total size of the combined notification and diagnostic logs can reach is 4 GB (4 x 1 GB).

Interpretation of diagnostic log file entries:

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

As an alternative to using **db2diag**, you can use a text editor to view the diagnostic log file on the machine where you suspect a problem to have occurred. The most recent events recorded are the furthest down the file.

Note: The administration notification (*instance_name*.nfy) and diagnostic (db2diag) logs grow *continuously* as single log files. When the **diagsize** database manager configuration parameter is set to a nonzero value, both the administration notification and the db2diag log files become a series of rotating log files (*instance_name.N*.nfy and db2diag.*N*.log) having a limited total size which is determined by the value of the **diagsize** configuration parameter.

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 file records.

```
2007-05-18-14.20.46.973000-240 1  I27204F655 2  LEVEL: Info 3
PID : 3228 4  TID : 8796 5  PROC : db2syscs.exe 6
INSTANCE: DB2MPP 7  NODE : 002 8  DB : WIN3DB1 9
APPHDL : 0-51 10  APPID: 9.26.54.62.45837.070518182042 11
```

```
AUTHID : UDBADM 12
EDUID : 8796 13  EDUNAME: db2agntp 14  (WIN3DB1) 2
FUNCTION: 15  Db2, data management, sqldInitDBCB, probe:4820
DATA #1 : 16  String, 26 bytes
Setting ADC Threshold to:
DATA #2 : unsigned integer, 8 bytes
1048576
```

Legend:

1. A timestamp and timezone for the message.

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

2. The record ID field. The recordID of the **db2diag** log files specifies the file offset at which the current message is being logged (for example, "27204") and the message length (for example, "655") for the platform where the Db2 diagnostic log was created.

3. The diagnostic level of the message. The levels are `Info`, `Warning`, `Error`, `Severe`, `Critical`, and `Event`.

4. The process ID

5. The thread ID

6. The process name

7. The name of the instance generating the message.

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

9. The database name

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

11. Identification of the application for which the process is working. In this example, the process generating the message is working on behalf of an application with the ID 9.26.54.62.45837.070518182042.

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

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

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

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

To identify more about a particular application ID, either:

- Use the **LIST APPLICATIONS** command on a Db2 server or LIST DCS APPLICATIONS on a Db2 Connect gateway to view a list of application IDs. From this list, you can determine information about the client experiencing the error, such as its database partition name and its TCP/IP address.
- Use the **GET SNAPSHOT FOR APPLICATION** command to view a list of application IDs.
- Use the **db2pd -applications -db <dbname>** command.

12 The authorization identifier.

13 The engine dispatchable unit identifier.

14 The name of the engine dispatchable unit.

15. The product name ("Db2"), component name (“data management”), and function name (“sqlInitDBCB”) that is writing the message (as well as the probe point (“4820”) within the function).

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

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

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

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

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

When the CALLED field is also present, <msgText> is an impact of the error returned by the CALLED function on the function logging a message (as specified in the FUNCTION field)

- CALLED This is the function that returned an error. It consists of:
  - <prodName> - The product name: "OS", "Db2", "Db2 Tools" or "Db2 Common"
  - <compName> - The component name ('-' in case of a system call)
  - <funcName> - The called function name
- OSERR This is the operating system error returned by the CALLED system call. (You can find it at the end of the same line as CALLED.) It consists of:
  - <errorName> - the system specific error name
  - <errno> - the operating system error number
- ARG This section lists the arguments of a function call that returned an error. It consists of:
  - <N> - The position of an argument in a call to the "called" function
  - <typeTitle> - The label associated with the Nth argument typename
  - <typeName> - The name of the type of argument being logged
  - <size> - The size of argument to be logged
- DATA This contains any extra data dumped by the logging function. It consists of:
  - <N> - The sequential number of data object being dumped
  - <typeTitle> - The label of data being dumped
  - <typeName> - The name of the type of data field being logged, for example, PD_TYPE_UINT32, PD_TYPE_STRING
  - <size> - The size of a data object

Interpreting the informational record of the db2diag log files:

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

An example of an informational record is as follows:

```
2006-02-09-18.07.31.059000-300 I1H917           LEVEL: Event
PID     : 3140                 TID  : 2864       PROC : db2start.exe
INSTANCE: DB2                  NODE : 000
FUNCTION: Db2, 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.

Setting the error capture level of the diagnostic log files:

The Db2 diagnostic (**db2diag**) log files are files that contain text information logged by Db2 database systems. This information is used for troubleshooting and much of it is primarily intended for IBM Software Support.

About this task

The types of diagnostic errors that are recorded in the **db2diag** log files are determined by the **diaglevel** database manager configuration parameter setting.

Procedure

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

  Look for the following variable:

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

- To change the value dynamically, use the **UPDATE DBM CFG** command.

  To change a database manager configuration parameter online:

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

  For example:

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

  where *X* is the notification level you want. If you are diagnosing a problem that can be reproduced, IBM Software Support personnel might suggest that you use **diaglevel** 4 while performing troubleshooting.

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 must 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 your Db2 server compete for resources?
- Where does authentication occur?

Most platforms have straightforward commands for retrieving resource information. However, you will rarely be required 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 generated data about the Db2 server 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 by the **diagpath** database manager configuration parameter).
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 v9.7.800.683 n130210 SQL09078
timestamp: 2013-03-15-10.32.37.894000
uname: S:Windows
comment: IP23428
process id: 7224
thread id: 6032
</Header>
<SystemInformation>
Number of Processors: 2
Processor Type: AMD64 Family 6 Model 44 Stepping 2
OS Version: Microsoft Windows Longhorn, Service Pack 1 (6.1)
Current Build: 7601
</SystemInformation>
<MemoryInformation>
<Usage>
Physical Memory:    8191 total,    2545 free.
Virtual Memory : 8388607 total, 8387728 free.
Paging File    :   16381 total,   11030 free.
Ext. Virtual   :       0 free.
</Usage>
</MemoryInformation>
<EnvironmentVariables>
M![CDATA[
[e] DB2PATH=D:\SQLLIB
[n] DB2INSTPROF=C:\ProgramData\IBM\DB2\db2build
[g] DB2_EXTSECURITY=YES
[g] DB2_COMMON_APP_DATA_PATH=C:\ProgramData
[g] DB2SYSTEM=JTANG
[g] DB2PATH=D:\SQLLIB
[g] DB2INSTDEF=DB2
[g] DB2ADMINSERVER=DB2DAS00
]]></EnvironmentVariables>
```

Correlating Db2 and system events or errors

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

While not always relevant to problem diagnosis, in many cases the best clue is readily available in the system logs. If you can correlate a reported system problem with Db2 errors, you will have often identified what is directly causing the Db2 symptom. Obvious examples are disk errors, network errors, and hardware errors. Not so obvious are problems reported on different machines, for example domain controllers which can affect connection time or authentication.

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

Here is some other information provided by system logs.
- Significant events such as when the system was rebooted
- Chronology of Db2 traps on the system (and errors, traps, or exceptions from other software that is failing)
- Kernel panics, out-of-filesystem-space, and out-of-swap-space errors (which can prevent the system from creating or forking a new process)

System logs can help to rule out crash entries in the **db2diag** log files as causes for concern. If you see a crash entry 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.

About SQL PL and PL/SQL error stack logging:

Starting in Db2 Version 10.5 Fix Pack 7, you can now determine the origin of run-time errors which occur within multiple nested layers of SQL routine invocations. There are two distinct features to help determine the source of an error in a list of active routines, including SQL and external routines, that were running at the time an SQL error took place.

The first error stack trace facility is intended to facilitate problem determination in production environments where the cause of a run-time error is unknown. It is activated at the database level.

pl_stack_trace
: This database configuration parameter logs SQL errors in procedural code to the `db2diag.log` file. Each entry contains a formatted representation of the complete call stack, showing the routine name and source line number for all active SQL and external routines on the stack. This allows easy identification of the code path at the time the error was detected. This facility does not require modification of the user application.

The second feature is a pair of utility functions. These functions can be used in application code, as part of an error reporting strategy, or for problem determination in a development environment. The functions are describes as follows:

DBMS_UTILITY.FORMAT_CALL_STACK
This function returns a formatted representation of the call stack, as it exists at the time the function is invoked. Each line of output describes one SQL or external routine. The lines are ordered from latest to oldest function calls.

DBMS_UTILITY.FORMAT_ERROR_BACKTRACE
This function returns produces output similar to that returned by FORMAT_CALL_STACK() function, but for the call stack as it existed at the time of the last error in a compiled SQL routine.

db2cos (callout script) output files:

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

The names for the **db2cos** scripts are in the form **db2cos_hang**, **db2cos_trap**, and so on. Each script behaves in a similar way except **db2cos_hang** which is called from the **db2fodc** tool.

The default **db2cos** scripts are found under the `bin` directory. On UNIX operating systems, this directory is read-only. You can copy the **db2cos** script file and all the relevant scripts to the `adm` directory if customization on all files is required. If a **db2cos** script is found in the `adm` directory, it is run; otherwise, the default script in the `bin` directory is run. For example, for data corruption issues, Db2 looks for a **db2cos_datacorruption** script. If there is a customized script in the `adm` directory, Db2 uses this customized script. Otherwise, the default **db2cos_datacorruption** script in the `bin` directory is used. For information on what scripts are called automatically and for which situation see, Collecting diagnosis information based on common outage problems.

For example, for data corruption issues, Db2 looks for a **db2cos_datacorruption** script. If there is a customized script in the `adm` directory, Db2 uses this customized script. Otherwise, the default **db2cos_datacorruption** script in the `bin` directory is used.

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

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

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

1. Trap file is created

2. Signal handler is called
3. db2cos script is called (depending on the db2cos settings enabled)
4. An entry is logged in the administration notification log
5. An entry is logged in the **db2diag** log file

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 have to collect further information, update the **db2cos** script with the additional commands.

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

The **db2cos** output files will contain different information depending on the commands specified in the **db2cos** script. If the default script is not altered, entries similar to the following will be displayed (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 files 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, 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, 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, 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, oper system services, sqloEDUCodeTrapHandler, probe:20
DATA #1 : Signal Number Recieved, 4 bytes
11
DATA #2 : Siginfo, 64 bytes
0x0FFFFFFFFFFFD5C0 : 0000 000B 0000 0000 0000 0009 0000 0000    ................
0x0FFFFFFFFFFFD5D0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5E0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
0x0FFFFFFFFFFFD5F0 : 0000 0000 0000 0000 0000 0000 0000 0000    ................
```

Dump files:

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

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

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

Note: For partitioned database environments, the file extension identifies the partition number. For example, the following entry indicates that the dump file was created by a Db2 process running on partition 10:

```
Dump File: /home/db2/sqllib/db2dump/6881492.2.010.dump.bin
```

First occurrence data capture information:

First occurrence data capture (FODC) collects diagnostic information about a Db2 instance, host or member when a problem occurs. FODC reduces the need to reproduce a problem to obtain diagnostic information, because diagnostic information can be collected as the problem occurs.

FODC can be invoked manually with the **db2fodc** command when you observe a problem or invoked automatically whenever a predetermined scenario or symptom is detected. After the diagnostic information has been collected, it is used to help determine the potential causes of the problem. In some cases, you might be able to determine the problem cause yourself, or involvement from IBM support personnel will be required.

Once execution of the **db2fodc** command has finished, the **db2support** tool must be executed to collect the resulting diagnostic files and prepare the FODC package to be submitted to IBM Support. The **db2support** command will collect the contents of all FODC package directories found or specified with the **-fodcpath** parameter. This is done to avoid additional requests, from IBM Support for diagnostic information.

Collecting diagnosis information based on common outage problems:

Diagnostic information can be gathered automatically in a first occurrence data collection (FODC) package as the problem that affects an instance, host, or member is occurring. User can use **db2fodc** command to collect performance data in manual FODC package on issues that do not trigger automatic FODC.

Automatic collection of diagnostic information

For specific errors, the database manager automatically collects diagnostic information in the First Occurrence Data Collection (FODC) package, which in turn may invoke one of the Db2 call-out scripts (COS).

To correlate the outage with the Db2 diagnostic logs and the other troubleshooting files, a diagnostic message is written to both the administration notification and the **db2diag** log files. The FODC package directory name includes the `FODC_` prefix, the outage type, the timestamp when the `FODC` directory was created, and the member or partition number where the problem occurred. The FODC package description file is placed in the new `FODC` package directory.

Table 2. Automatic FODC types and packages

Package	Description	Script executed
FODC_Trap_*timestamp_ memberNumber*	An instance wide trap has occurred	**db2cos_trap** (.bat)
FODC_Panic_*timestamp_ memberNumber*	Engine detected an incoherence and decided not to continue	**db2cos_trap** (.bat)
FODC_BadPage_*timestamp _memberNumber*	A Bad Page has been detected	**db2cos_datacorruption** (.bat)
FODC_DBMarkedBad_ *timestamp_ memberNumber*	A database has been marked bad due to an error	**db2cos** (.bat)
FODC_[Index|Data|Col] Error_directory _timestamp_PID_EDUID _memberNumber	An EDU wide index error occurred.	db2cos_*[index|data|col] error*_long(.bat) or db2cos_*[index|data|col] error*_short(.bat)
FODC_Member_*timestamp _memberNumber*	A member or partition has failed or has received a kill signal	**db2cos_member** (.bat)

Manual collection of diagnostic information

You use the **db2fodc** command manually when you suspect a problem is occurring. Problem scenarios that you can collect diagnostic data for include apparent system hangs, performance issues, or when an upgrade operation or instance creation did not complete as expected. When the **db2fodc** command is run manually, a new FODC package directory is created. The FODC package directory name includes the `FODC_` prefix, the problem scenario, the timestamp when the `FODC` directory was created, and the member(s) or partition number(s) where FODC was performed.

Table 3. Manual FODC types and packages

Package	Description	Script executed
FODC_Clp_*timestamp_ member*	User invoked **db2fodc -clp** to collect environment and configuration related information, used to troubleshoot problems related to instance creation.	**db2cos_clp** script (.bat)
FODC_Connections_ *timestamp_member*	User invoked **db2fodc -connections** to collect connection-related diagnostic data, used to diagnose problems such as sudden spikes in the number of applications in the executing or compiling state or new database connections being denied.	**db2cos_threshold** script (.bat)
FODC_Cpu_*timestamp_ member*	User invoked **db2fodc -cpu** to collect processor-related performance and diagnostic data, used to diagnose problems such as high processor utilization rates, a high number of running processes, or high processor wait times.	**db2cos_threshold** script (.bat)
FODC_Hang_*timestamp_ memberList*	User invoked **db2fodc -hang** to collect data for hang troubleshooting (or severe performance)	**db2cos_hang** (.bat)
FODC_Memory_*timestamp _member*	User invoked **db2fodc -memory** to collect memory-related diagnostic data, used to diagnose problems such as no free memory available, swap space being used at a high rate, excessive paging or a suspected a memory leak.	**db2cos_threshold** script (.bat)
FODC_Perf_*timestamp_ memberList*	User invoked **db2fodc -perf** to collect data for performance troubleshooting	**db2cos_perf** (.bat)
FODC_Preupgrade_ *timestamp_member*	User invoked **db2fodc -preupgrade** to collect performance related information before a critical upgrade or update such as upgrading an instance or updating to the next fix pack	**db2cos_preupgrade** (.bat)

Table 3. Manual FODC types and packages (continued)

Package	Description	Script executed
Scripts located in FODC_-fodcerror *FODC_[Index|Data|Col] Error_directory_ timestamp_PID_EDUID_ memberList*	User could issue **db2fodc -fodcerror** *FODC_[Index|Data|Col] Error_directory* [**basic** | **full**] (default is basic) to invoke the **db2dart** commands in the scripts. On partitioned database environments, use **db2_all** "<<+*node*#< **db2fodc -fodcerror** *FODC_[Index|Data|Col] Error_directory* [**basic** | **full**]". The *node*# is the last number in the *FODC_[Index|Data|Col] Error_directory* directory name. An absolute path is required when using **db2fodc -fodcerror** *FODC_[Index|Data|Col] Error_directory* with the **db2_all** command.	db2cos_*[index|data|col] error*_long(.bat) or db2cos_*[index|data|col] error*_short(.bat)

First occurrence data capture configuration:

First occurrence data capture configuration (FODC) behaviour, including the path used to store the FODC package, is controlled by the *DB2FODC* registry variable, which can be set persistently with the **db2set** command or changed dynamically (in-memory only) through the **db2pdcfg** command. FODC behavior can also be customized by updating the call-out scripts (COS) invoked during FODC.

Each partition or member in the instance has its own FODC settings, and you can control how FODC takes place at the partition or member level. If FODC settings exist both at the member or partition level and at the instance level, the member or partition level settings override the instance level settings. For manual FODC, settings can also be overridden by command line parameters you specify, such as the **-fodcpath** parameter. In partitioned or Db2 pureScale® database environments, if you specify a list of members or partitions for manual FODC, the settings for the first member or partition specified are used.

Persistent settings made with the **db2set** command do not become effective until the instance is recycled; dynamic settings made with the **db2pdcfg** command are effective immediately and remain effective in memory until the instance is recycled.

To help you control how FODC packages are handled, several *DB2FODC* registry variable settings are available, but not all settings are available on all platforms. You can control the following behaviors through the *DB2FODC* registry variable:

- Where the generated FODC packages are stored (with the *FODCPATH* setting)
- Whether core dump files are generated or not (with the *DUMPCORE* setting)
- How big core dump files can become (with the *CORELIMIT* setting)
- Where the generated core files are stored (with the *DUMPDIR* setting)

FODC by default invokes a **db2cos** call-out script to collect diagnostic information when the database manager cannot continue processing due to a panic, trap, segmentation violation or exception. To help you control the call-out script that is invoked during FODC, several *COS* parameter settings are available. You can control the following behaviors through the *COS* parameter of the *DB2FODC* registry variable:

- Whether the **db2cos** script is invoked when the database manager cannot continue processing (with the *ON* and *OFF* setting; the default is *ON*)
- How often the **db2cos** script checks the size of the output files generated (with the *COS_SLEEP* setting)
- How long FODC should wait for the **db2cos** script to finish (with the *COS_TIMEOUT* setting)
- How often the **db2cos** script is invoked during a database manager trap (with the *COS_COUNT* setting)
- Whether the **db2cos** script is enabled when the SQLO_SIG_DUMP signal is received (with the *COS_SQLO_SIG_DUMP* setting)

FODC package directory settings (FODCPATH)

FODC packages can result in the generation of large volumes of diagnostic data that require space to store and can impose a significant processor usage on the system. You can control what directory path FODC sends diagnostic data to, so that you can pick a directory path with sufficient free space available.

The following order is used to determine what FODC path to use:

Automatic FODC

FODCPATH registry variable setting
The **FODCPATH** parameter for the **DB2FODC** registry variable can be set at the member or partition level and at the instance level. FODC uses the **FODCPATH** parameter setting for each partition or member, if set. If a partition or member level setting does not exist, the instance level setting is used.

No FODC path settings
By default, if you do not specify any **FODCPATH** setting at either the member or instance level, FODC sends diagnostic information to the current diagnostic directory path (**diagpath** or **alt_diagpath**).

Manual FODC

db2fodc -fodcpath command parameter option
When manually invoking the **db2fodc** command, you can indicate the location where the FODC package directory is created by specifying the **-fodcpath** parameter option together with the command. If you specify the **-fodcpath** parameter with a valid path name, the FODCpackage directory is created in that path.

FODCPATH registry variable setting
If you do not specify the **-fodcpath** parameter with the **db2fodc** command, and you specified a list of partitions or members, the **db2fodc** command uses the **FODCPATH** parameter setting for the **DB2FODC** registry variable of the first partition or member from the list specified. If the value for that **FODCPATH** parameter is not set, **db2fodc** uses the instance level **FODCPATH** setting. If you do not specify the **-fodcpath** parameter and do no specify a list of partitions or members, the **db2fodc** command first tries to use the **FODCPATH** parameter setting for the current partition or member; if not set, the instance level setting is used.

No FODC path settings

By default, if you do not specify any FODC path, first occurrence data capture sends diagnostic information to the current diagnostic directory path (**diagpath** or **alt_diagpath**).

Assume that you have a partitioned database environment with 3 members or partitions (0, 1, and 2). The following example shows how to set the FODC path persistently at the instance level for all 3 partitions or members using the **db2set** command:

```
db2set DB2FODC=FODCPATH=/home/hotel49/juntang/FODC
```

FODC path settings can also be performed persistently at the member level for each member, overriding the instance level setting. To make these settings effective, the instance must be recycled. For example, to change the FODC path on member 0, issue the following command:

```
db2set DB2FODC=FODCPATH=/home/hotel49/juntang/FODC/FODC0 -i juntang 0
```

If you now want to change the FODC path dynamically on member 1 and member 2, you use the following **db2pdcfg** commands. These settings are effective immediately and remain in memory until the instance is recycled.

```
db2pdcfg -fodc FODCPATH=/home/hotel49/juntang/FODC/FODC1 -member 1

db2pdcfg -fodc FODCPATH=/home/hotel49/juntang/FODC/FODC2 -member 2
```

If you want to know what the current FODC settings are for each member or partition in a system, you can use the **db2pdcfg -fodc -member all** command (in the example, output is abridged and only the FODC path output is shown):

```
Database Member 0
FODC package path (FODCPATH)= /home/hotel49/juntang/FODC/FODC0/

Database Member 1
FODC package path (FODCPATH)= /home/hotel49/juntang/FODC/FODC1/

Database Member 2
FODC package path (FODCPATH)= /home/hotel49/juntang/FODC/FODC2/
```

Customized data collection

The behavior of data collection by **db2fodc -hang** and **db2fodc -perf** is also controlled by parameters defined in the TOOL OPTIONS section of the Db2 call-out script that is invoked during FODC. These parameters can be customized by changing the script that is executed during FODC.

To customize the data collection on UNIX systems, copy the script placed in `/bin/db2cos_`*`symptom`* to `/adm/db2cos_`*`symptom`*, where *symptom* is either `hang` or `perf`. Once in this new directory, modify the script as you like. On Windows systems, simply modify the default script `\bin\db2cos_`*`symptom`*`.bat`. On UNIX systems, **db2fodc** first tries to execute the script in `/adm/db2cos_`*`symptom`*, and, if it is not found, executes the original script in `/bin/db2cos_`*`symptom`*. On Windows systems, the script `\bin\db2cos_`*`symptom`*`.bat` is always executed.

Data collected as part of FODC:

First occurrence data capture (FODC) results in the creation of a FODC package directory and subdirectories where diagnostic information is collected. The parent package directory, subdirectories and files that get collected are collectively known as a FODC package.

Files containing diagnostic information that are collected by FODC

FODC collects diagnostic information from a number of sources. The exact diagnostic information captured by FODC depends on the type of problem encountered and might include:

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.

  Note: When the **diagsize** database manager configuration parameter is set to a nonzero value, the single administration notification log file behavior (*instance_name*.nfy) will be changed to a rotating log behavior (*instance_name.N*.nfy).

Db2 diagnostic log (db2diag log file)

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

  Note: When the **diagsize** database manager configuration parameter is set to a nonzero value, the single diagnostic log file behavior (a single db2diag.log file) will be changed to a rotating log behavior (db2diag.*N*.log).

Db2 administration server (DAS) diagnostic log (db2dasdiag.log)

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

Db2 event log (db2eventlog.xxx, where xxx is the database partition number)

- Operating system: All

- Default location: Located in the directory specified by the **diagpath** database manager configuration parameter
- Created automatically when the instance is created.
- The Db2 event log file is a circular log of infrastructure-level events occurring in the database manager. The file is fixed in size, and acts as circular buffer for the specific events that are logged as the instance runs. Every time you stop the instance, the previous event log will be replaced, not appended. If the instance traps, a db2eventlog.XXX.crash file is also generated. These files are intended for use by IBM Software Support.

Db2 callout script (db2cos) output files

- Operating system: All
- Default location: Located in the directory specified by the **diagpath** database manager configuration parameter
- If db2cos scripts are executed as a consequence of an FODC outage, db2cos output files will be placed under the FODC directory that was created in the location 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, such as operating system commands and other Db2 diagnosing tools. For more details on the tools that are executed with the db2cos script, open the script file in a text editor.
- The db2cos script is shipped under the `bin/` directory. On UNIX, this directory is read-only. To create your own modifiable version of this script, copy the db2cos script to the `adm/` directory. You are free to modify this version of the script. If the script is found in the `adm/` directory, it is that version that is run. Otherwise, the default version in the `bin/` directory is run.

Dump files

- Operating system: All
- Default location: Located in the directory specified by the **diagpath** database manager configuration parameter
- If these files are dumped during an FODC outage, they will be placed under the FODC directory.
- Created automatically when particular problem scenarios arise.
- For some error conditions, extra information is logged in binary files named after the failing process ID. These files are intended for use by IBM Software Support.

Trap files

- Operating system: All
- Default location: Located in the directory specified by the **diagpath** database manager configuration parameter
- If these files are dumped during an FODC outage, they will be placed under the FODC directory.

- 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
- If these files are dumped during an FODC outage, they will be placed under the FODC directory.
- Created by the operating system when the Db2 instance terminates abnormally.
- Among other things, the core image will include most or all of the memory allocations of Db2, which may be required for problem descriptions.

FODC package path and contents

FODC creates the FODC package directory in the FODC path specified. You specify the FODC path through the **FODCPATH** registry variable setting or the **db2fodc -fodcpath** command parameter option. If you do not specify any FODC path, first occurrence data capture sends diagnostic information to the current diagnostic directory path (**diagpath** or **alt_diagpath**). A **db2diag** log file diagnostic message is logged to identify the directory name used for FODC. The capture of diagnostic information can generate a significant volume of diagnostic data, depending on what parameters are specified, and enough space must be available in the directory path where FODC stores diagnostic information. To avoid a scenario where FODC fills all the available space in the file system and impacts your data server, it is recommended that you specify a FODC path where FODC can store the diagnostic data.

For automatic FODC, a package is collected for the member or partition where the problem is occurring; if the problem is occurring on multiple members, multiple packages are collected in separate FODC package directories. The FODC package directory follows the naming convention FODC*_outageType_timestamp_member_number*, where *outageType* is the problem symptom, *timestamp* is the time of FODC invocation, and *member_number* is the member or partition number where the problem occurred. For example, when a trap occurs on member 1, FODC might automatically create a package named like `FODC_Trap_ 2010-11-17-20.58.30.695243_0001`.

For manual FODC, a package is collected for the member(s) or partition(s) you specify. The naming convention for the FODC package directory is FODC*_manualOutageType_timestamp_memberList*, where *manualOutageType* is the problem symptom, *timestamp* is the time of FODC invocation, and *memberList* is a list of the members or partitions where the problem occurred. For example, the manually issued command **db2fodc -hang -basic -member 1,2,3 -db sample** creates a manual FODC package for members 1,2 and 3, and might be named like `FODC_hang_ 2010-11-17-20.58.30.695243_0001.0002.0003`.

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

- DB2CONFIG containing Db2 configuration output and files

- DB2PD containing **db2pd** output or output files
- DB2SNAPS containing Db2 snapshots
- DB2TRACE containing Db2 traces
- OSCONFIG containing operating system configuration files
- OSSNAPS containing operating system monitor information
- OSTRACE containing operating system traces

Not all of these directories might exist, depending on your FODC configuration and the outage type for which the **db2fodc** command is run.

FODC sends the following diagnostic information to the FODC package directory:

db2fodc -clp collects the following information:

- Operating system information
- Instance and database configuration information

db2fodc -hang collects the following information:

db2fodc -hang collects the following info:

- Basic operating system information. The problem could be due to OS level, patches, and so on.
- Basic Db2 configuration information.
- Operating system monitor information: vmstat, netstat, iostat, and so on.
  - 2 iterations at least: with timestamps saved
- Partial call stacks: Db2 stack traces of top CPU agents.
- Operating system trace: trace on AIX.
- Diagnostic information collected by **db2pd**.
- Db2 trace.
- Full Db2 call stacks.
- Second round of Db2 configuration information.
  - Including second Db2 trace collection.
- Snapshot information: **db2 get snapshot for** database, applications, tables, and so on.
  - Information will be collected per node in case of multiple logical nodes.

db2fodc -perf monitors the system possibly collecting the following information:

- Snapshots
- Stacktraces
- Virtual Memory (Vmstat)
- Input/Output information (Iostat)
- traces
- Some other information depending on the case. See the script for more details.

db2fodc -fodcerror *FODC_[Index|Data|Col]Error_directory* collects the following data that is related to:

- an index error (*FODC_IndexError_directory*),
- a database manager error (*FODC_DataError_directory*), or
- a column-organized table error (*FODC_ColError_directory*).

(*FODC_DataError_directory* and *FODC_ColError_directory* were added in the Db2 Cancun Release.)

- Basic Mode
  - db2cos_*[index|data|col]error*_short(.bat) script is run. See script for additional details.
  - For db2cos_indexerror_short(.bat), if applicable **db2dart** commands exist in the script, the **db2dart /DD**, **db2dart /DI**, or both data formatting actions are run with number of pages limited to 100.
  - For db2cos_dataerror_short(.bat), if applicable **db2dart** commands exist in the script, the **db2dart /DD** data formatting action is run with number of pages limited to 100.
  - For db2cos_colerror_short(.bat), if applicable **db2dart** commands exist in the script, the **db2dart /DC** data formatting action is run with number of pages limited to 100.
- Full Mode
  - db2cos_*[index|data|col]error*_short(.bat) and db2cos_*[index|data|col]error*_long(.bat) scripts are run. See scripts for additional details.
  - For db2cos_indexerror_short|long(.bat):
    - If applicable **db2dart** commands exist in the script db2cos_indexerror_short(.bat), the **db2dart /DD**, **db2dart /DI**, or both data formatting actions are run with number of pages limited to 100.
    - If applicable **db2dart** commands exist in the script db2cos_indexerror_long(.bat), the **db2dart /DD**, **db2dart /DI**, or both data formatting actions are run with no limit to the number of pages.
    - If applicable **db2dart** commands exist in the db2cos_indexerror_long(.bat) script, the **db2dart /T** command is run. This command requires the database be offline.
  - For db2cos_dataerror_short|long(.bat):
    - If applicable **db2dart** commands exist in the script db2cos_dataerror_short(.bat), the **db2dart /DD** data formatting action is run with number of pages limited to 100.
    - If applicable **db2dart** commands exist in the script db2cos_dataerror_long(.bat), the **db2dart /DD** data formatting action is run with no limit to the number of pages.
    - If applicable **db2dart** commands exist in the db2cos_dataerror_long(.bat) script, the **db2dart /T** command is run. This command requires the database be offline.
  - For db2cos_colerror_short|long(.bat):
    - If applicable **db2dart** commands exist in the script db2cos_colerror_short(.bat), the **db2dart /DC** data formatting action is run with number of pages limited to 100.
    - If applicable **db2dart** commands exist in the script db2cos_colerror_long(.bat), the **db2dart /DC** data formatting action is run with no limit to the number of pages.
    - If applicable **db2dart** commands exist in the db2cos_colerror_long(.bat) script, the **db2dart /T** command is run. This command requires the database be offline.

db2fodc -preupgrade collects the following information:

- Operating system information
- Instance and database configuration information, such as output of the **db2level** command, environment variables, output of the **db2 get dbm cfg** command, and the db2nodes.cfg file
- System catalog data and statistics, such as optimizer information collected by the **db2support -d** *dbname* **-c -s -cl 0** command
- Operating system monitoring data, such as output of the **netstat -v** and **ps -elf** commands
- System files
- Package information, as returned by the Db2 LIST PACKAGES FOR SCHEMA *schema-name* SHOW DETAIL command for all schema names
- Any FODC_Preupgrade directories found in db2dump/. These directories contain information such as performance data, top dynamic SQL queries, and explain plans
- The logfile from the **db2ckupgrade** command in /tmp/ db2ckupgrade.log.*processID*, if it exists
- Output from the **db2prereqcheck** command

The following diagnostic information is also included when you specify the members on which to collect:

- Snapshots (after turning on all monitor switches)
- The **db2pd** command output for the -everything, -agents, -applications, -mempools, and -fcm parameters
- The dynamic SQL statements used most frequently
- The query plans for SQL statements
- The explain plans for static packages

Manual **db2fodc** command invocation results in the creation of a log file named db2fodc_*symptom*.log in the FODC_*symptom* directory, where *symptom* is one of the collection types, such as hang or perf. Inside this file, the **db2fodc** command also stores status information and metadata describing the FODC package inside the FODC subdirectory. This file contains information about the type of FODC, the timestamp of the start and end of data collection, and other information useful for the analysis of the FODC package.

Automatic FODC data generation:

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

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

Trap files contain information such as:

- The amount of free virtual storage
- Values associated with the product's configuration parameters and registry variables at the time the trap occurred
- Estimated amount of memory used by the Db2 product at the time of the trap
- Information that provides a context for the outage

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

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

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

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

Db2 audit log ("`db2audit.log`")

- Operating system: All
- Default location:
  - Windows: Located in the `$DB2PATH\`*`instance_name`*`\security` directory
  - Linux and UNIX: Located in the `$HOME\sqllib\security` directory, where $HOME is the instance owner's home directory
- Created when the **`db2audit`** facility is started.
- Contains audit records generated by the Db2 audit facility for a series of predefined database events.

Db2 governor log ("*`mylog.x`*", where *x* is the number of database partitions on which the governor is running)

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

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

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

Graphical tools using First Occurrence Data Capture (FODC):

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

Data Warehouse Center IWH2LOGC.log file

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

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

Data Warehouse Center IWH2LOG.log file

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

Data Warehouse Center IWH2SERV.log file

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

Information Catalog Center tag file EXPORT log

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

Information Catalog Center tag file IMPORT log

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

Internal return codes:

There are two types of internal return codes: ZRC values and ECF values. These are return codes that will generally only be visible in diagnostic tools intended for use by IBM Software Support.

For example, they are displayed in Db2 trace output and in the **db2diag** log files.

ZRC and ECF values basically serve the same purpose, but have slightly different formats. Each ZRC value has the following characteristics:
- Class name
- Component
- Reason code
- Associated SQLCODE
- SQLCA message tokens
- Description

However, ECF values consist of:
- Set name
- Product ID
- Component
- Description

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

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

```
2006-02-13-14.34.35.965000-300    I17502H435       LEVEL: Error
PID     : 940                TID  : 660    PROC : db2syscs.exe
INSTANCE: Db2                NODE : 000     DB   : SAMPLE
APPHDL  : 0-1433               APPID: *LOCAL.DB2.050120082811
FUNCTION: Db2, data protection, sqlpsize, probe:20
RETCODE : ZRC=0x860F000A=-2045837302=SQLO_FNEX "File not found."
          DIA8411C A file "" could not be found.
```

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

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

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

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

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

Identifier:
        SQLO_FNEX
        SQLO_MOD_NOT_FOUND
```

```
Identifier (without component):
        SQLZ_RC_FNEX

Description:
        File not found.

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

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

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

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

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

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

ECF value to map: 0x90000076 (-1879048074)

ECF Set :
        setecf (Set index : 1)
Product :
        Db2 Common
Component:
        OSSe
Code:
        118 (0x0076)

Identifier:
        ECF_LIB_CANNOT_LOAD

Description:
        Cannot load the specified library
```

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

For a full listing of the ZRC values, use the **db2diag -rc zrc** command and for a full listing of the ECF values, use the **db2diag -rc ecf** command.

Platform-specific error log information:

There are many other files and utilities available outside of Db2 to help analyze problems. Often they are just as important to determining root cause as the information made available in the Db2 files.

The other files and utilities provide access to information contained in logs and traces that is concerned with the following areas:
- Operating systems
- Applications and third-party vendors
- Hardware

Based on your operating environment, there might be more places outside of what has been described here, so be aware of all of the potential areas you might have 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 error report logs are accessed using the **/usr/bin/errpt -a** command; the system logs are enabled using the `/etc/syslog.conf` file
- 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. See 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 must identify what hardware diagnostic files and utilities might be available for piece of hardware in your environment. An example of such a case is when a bad page, or a corruption of some type is reported by Db2. Usually this is reported due to a disk problem, in which case the hardware diagnostics must be investigated. See your hardware documentation and support material to determine what further information is available.

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

In summary, to completely understand and evaluate a problem, you might have 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 require, but you should still be aware of any information outside of this that might help the investigation.

System core files (Linux and UNIX):

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

The core file is named "core", and is placed in the directory specified by the **diagpath** database manager configuration parameter, by default, unless otherwise configured using the values in the **DB2FODC** registry variable. Note that system core files are distinct from Db2 trap files.

Core file control settings

Core files can result in the generation of large volumes of diagnostic data that require space to store and can impose a significant processor usage on the system. To help you control how core files are handled, several **DB2FODC** registry variable settings are available. You can make **DB2FODC** registry variable settings persistently with the **db2set** command or change them dynamically (in-memory only) through the **db2pdcfg** command. Persistent settings made with the **db2set** command do not become effective until the instance is recycled; dynamic settings made with the **db2pdcfg** command are effective immediately and until the instance is recycled.

You can control the following core file behaviors through the **DB2FODC** registry variable:

- Whether core files are generated or not (with the **DUMPCORE** setting)
- How big core files can become (with the **CORELIMIT** setting)
- Where the generated core files are stored (with the **DUMPDIR** setting)

As a rule, a core file can become as large as the amount of physical memory installed on the machine where the core file is generated. For example, if your data server has 64 GB of physical memory, there must be at least 64 GB of space available in the directory path where the core file will be stored. It is possible to limit the size of the core file, but it is recommended that you configure core file behaviour to point to a file system with enough space available instead. If you must limit the amount of space a core file can use, ensure that the amount of space available is at least as great as the amount of physical memory on the machine or you risk truncating the core file and losing diagnostic information. For example, to limit the amount of space available for core file generation to 64 GB and to redirect core files to `/tmp` persistently, issue the following command. The settings will become effective only after the instance is recycled.

```
db2set DB2FODC="CORELIMIT=64000000000 DUMPDIR=/tmp"
```

When a core file is generated, it can impose a significant processor usage on the system, which in turn can affect system availability. If the performance impact to system availability during core file generation is unacceptable, you can disable core file generation, but it is recommended that you do not disable it permanently. Core files contain diagnostic information that can be required in order to troubleshoot a problem successfully. If diagnostic information is not available because core file generation was turned off permanently, troubleshooting a problem with your data server might become impossible. For example, to turn core file generation off dynamically, effective immediately and until the instance is recycled, issue the following command:

```
db2pdcfg DB2FODC="DUMPCORE=OFF"
```

Accessing system core file information (Linux and UNIX):

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

Before you begin

- You must have the **dbx** command installed. This command is operating system-specific: on AIX and Solaris, use **dbx**; on HP-UX, use **gdb** or **wdb**, and on Linux use **gdb**.

- On AIX, ensure that the full core option has been enabled using the **chdev** command or smitty.

Procedure

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

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

   ```
   dbx program_name core_filename
   ```

   program_name is the name of the program that terminated abnormally, and *core_filename* is the name of the file containing the core file dump. The *core_filename* parameter is optional. If you do not specify it, the default name "core" is used.
2. Examine the call stack in the core file. Information about how to do this can be obtained by issuing `man dbx` from a UNIX command prompt
3. To end the **dbx** command, type **quit** at the **dbx** prompt.

Example

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 *) pcAddress[i]);
   ```
3. The name of the function that caused the core dump is "freeSegments". Enter **where** at the dbx prompt to display the program path to the point of failure.

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

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

Accessing event logs (Windows):

This task describes how to access the Windows event logs.

About this task

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

Procedure

View the event logs using the Windows Event Viewer. The method used to open the viewer will differ, depending on the Windows operating system you are using. For example, to open the Event Viewer on Windows 7, click **Start** > **Control Panel**. Select **System and Maintenance**, and then select **Administrative Tools**. Double-click **Event Viewer**.

Exporting event logs (Windows):

This task describes how to export Windows event logs.

About this task

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

Procedure

Export the event logs from the Windows event viewer.
- You can load the log-file format (*.evt) data back into an event viewer (for example, on another workstation). This format is easy to work with since you can use the viewer to switch the chronology order, filter for certain events, and advance forwards or backwards.
- You can open the text (*.txt) or comma-delimited (*.csv) format logs in most text editors. They also avoid a potential problem with respect to timestamps. When you export event logs in .evt format, the timestamps are in Coordinated Universal Time and get converted to the local time of the workstation in the viewer. If you are not careful, you can overlook key events because of time zone differences. Text files are also easier to search.

Accessing the Dr. Watson log file (Windows):

This task describes how to access the Dr. Watson™ log files on Windows systems.

About this task

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.

Procedure

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

Configuring the System error or event log (syslog):

Syslog is a standard for computer message logging and integrates log data from many different types of systems into a central repository.

Authorization

Root user authority is required on UNIX operating systems.

Description

The program of the syslog is syslogd, that is, syslog daemon. The configuration file `/etc/syslog.conf` is used to control the output of syslogd. The user has to configure the log configuration file (`/etc/syslog.conf`) and each line in the configuration file must consist of the first two parts below:

1. A selector to determine the log message priorities which is the *facility.priority* pair.
2. A log destination (file path) for the above selector.
3. Rotation (optional)

The *facility* must be one of the values from the following list:
- kern - kernel messages
- user - random user-level messages (recommended for the **`db2audit extract`** command)
- mail - mail system messages
- daemon - system daemons
- auth - security/authorization messages (recommended for the **`db2audit extract`** command)
- syslog - messages generated internally by syslogd
- lpr - line printer subsystem
- news - news subsystem
- uucp - uucp subsystem
- cron - clock daemon
- caa - Cluster aware AIX subsystem
- local0 ~ local7 - reserved for local use (recommended for the **`db2audit extract`** command)
- * - (all facilities- used only in the configuration file and not in the commands or API)

The *priority* must be one of the values from the following list (from high to low) :
- emerg or panic - system is unusable
- alert - action must be taken immediately
- crit - critical conditions
- err or error - error conditions
- warn or warning - warning conditions
- notice - normal but significant condition
- info - informational
- debug - debug-level messages

Syslog messages are logged usually in the format:

date time hostname facility:priority username: message_body.

All items before the *message_body* are metadata, for example:

Oct 10 12:05:23 hotel37 mail:err newton: The user newton just got a mail error.

Examples

The following example shows the *facility.priority* sample configuration lines in the `/etc/syslog.conf` file:

```
user.info  /var/log/db2/user_messages.log
```

User messages at `info` or higher priority go to the `/var/log/db2/user_messages.log`.

```
mail.crit  /dev/console
```

Mail messages at `crit` or higher priority go to the console.

```
 *.debug /var/log/all_messages.log
```

All facilities at debug or higher priority go to`/var/log/all_messages.log`.

```
auth.warning @host123.torolab.ibm.com
```

Authorization messages at `warning` or higher priority are forwarded by the local syslog daemon (syslogd) to the syslog daemon (syslogd) on host123 machine.

Enabling the system error or event log (AIX):

The system error or event logs (syslog) are one set of diagnostic files that track activity and failures. On AIX systems, you need to manually enable the system logs.

About this task

The syslog content is not as comprehensive as the `db2diag.log` file, but you can find valuable diagnostic information there when the database manager is unable to write to the `db2diag.log` file or when there are errors in the Tivoli SA MP and RSCT subsystems.

You need to set up the log configuration file (`/etc/syslog.conf`) in order to enable system error logging on AIX. (On Linux, system error logging is enabled by default.)

Procedure

To enable system error or event logging, perform the following steps on each server:

1. Ensure that the syslog target file exists (assuming `syslog.out` is defined as the target file)
   ```
   touch /var/log/syslog.out
   ```
2. Restart the syslog daemon (syslogd) to pick up the changes to its config file (/etc/syslog.conf)
   ```
   refresh -s syslogd
   ```

What to do next

To access the syslog output file, use the following command: **`/usr/bin/errpt -a`**

Trap files:

Db2 generates a trap file if it cannot continue processing because of a trap, segmentation violation, or exception.

All signals or exceptions received by Db2 are recorded in the trap file. The trap file also contains the function sequence that was running when the error occurred. This sequence is sometimes referred to as the "function call stack" or "stack trace." The trap file also contains additional information about the state of the process when the signal or exception was caught.

A trap file is also generated when an application is forced to stop while running a fenced threadsafe routine. The trap occurs as the process is shutting down. This is not a fatal error and it is nothing to be concerned about.

The files are located in the directory specified by the **`diagpath`** database manager configuration parameter.

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

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

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

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

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

Note: If you are sending Windows trap files to IBM Support, ensure that you format your trap files before sending them to IBM Support in order to speed up the problem determination process. For information on how to format Windows trap files, see Formatting trap files (Windows)

Formatting trap files (Windows):

You can format trap files (*.TRP) with a command called **`db2xprt`**. It formats the Db2 database binary trap files into a human readable ASCII file.

About this task

The **`db2xprt`** 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.

Example

If a trap file called "DB30882416.TRP" had been produced in your directory specified by the **diagpath** database manager configuration parameter, you could format it as follows:

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

Checking archive log files with the db2cklog tool

Checking your archive log files ensures that known good log files are available in case a rollforward recovery becomes necessary and that the recovery operation does not fail because of a problem with a log file. The information in this topic tells you how to check log files with the **db2cklog** tool, and what to do if a log file does fail validation.

Before you begin

You need to have read permission on the archive log files, so that the **db2cklog** tool can read the log files and perform its checks. Only log files that are closed, such as archive log files, can be validated successfully. If you run the tool on a log file that is still active, the tool cannot check that file accurately and you will receive a warning to let you know that the file is still active.

About this task

The **db2cklog** tool reads either single log files or a range of numbered log files and performs checks on the internal validity of the files. Log files that pass validation without any error messages or warnings are known good files and you can use them during a rollforward recovery operation. If an archive log file fails validation with an error message or if a warning is returned, then you must not use that log file during rollforward recovery. An archive log file that fails validation cannot be repaired and you should follow the response outlined in this task for what to do next.

Checking your archive log files is useful in the following scenarios:

- Immediately before a rollforward operation is started: If it becomes necessary to perform a rollforward recovery operation, you can first run the **db2cklog** tool against all the archive log files that are required to perform the rollforward operation to ensure that the log files are valid. Running the **db2cklog** tool against the log files beforehand helps avoid a situation where a recovery operation fails partway through because of a problem with a log file, necessitating a follow-on recovery operation.
- Every time a log file is closed and copied to the log archive directory: As part of your day-to-day operations, archive log files can be checked as an added precaution to make sure that known good log files are always available. With this preventive measure, you know right away whether you need to locate a copy of a log file or if a full database backup to establish a new recovery point is needed. This helps to reduce any delay in the event that a rollforward recovery becomes necessary.

Procedure

To check your archive log files, you issue the **db2cklog** command from the command line and include the log file or files you want checked. Note that you do not specify full log file names with the **db2cklog** command but only the numeric identifiers that are part of the log file names. The numeric identifier of the

S0000001.LOG log file is 1, for example; to check that log file, you specify db2cklog 1. If the archive log files are not in the current directory, include the relative or absolute path to the log files with the optional **ARCHLOGPATH** parameter.

1. If you want to check the validity of a single archive log file, you specify the numeric identifier of that log file as *log-file-number1* with the command. For example, to check the validity of the S0000000.LOG log file in the /home/amytang/tests directory, you issue the command db2cklog 0 ARCHLOGPATH /home/amytang/tests.
2. If you want to check the validity of a range of archive log files, you include the first and last numeric identifier of that range with the command (from *log-file-number1* to *log-file-number2*). All log files in the range are checked, unless the upper end of the range specified with *log-file-number2* is numerically lower than the beginning of the range (specified with *log-file-number1*). In that case, only *log-file-number1* is checked. For example, to check the validity of the log files ranging from S0000000.LOG to S0000005.LOG in the /home/nrichers/tests directory, you issue the command db2cklog 0 TO 5 ARCHLOGPATH /home/nrichers/tests

Results

The **db2cklog** tool will return a return code of zero for any file that passes validation. If a range of numbered archive log files is specified, the **db2cklog** tool will read each file in sequence, perform its checks and issue a return code for each file. The tool stops at the first error it encounters, even if a range of log files was specified and there are additional files the tool has not yet checked. The DBT message that is returned when an error is found can provide you with some more information about why an archive log file failed validation, but you cannot fix an invalid log file. If you receive a DBT warning message that a log file might still be active but know for certain that the file is an archive log file, then you should treat the archive log file as invalid and follow the response for what to do next outlined in this task.

Example

The following example shows the typical output of the **db2cklog** command as it parses a log file, in this case S0000002.LOG. This file passes validation with a return code of zero.

```
$ db2cklog 2
     ____________________________________________________________________

                    _____     D B 2 C K L O G     _____

                           Db2 Check Log File tool
                                I    B    M


        The db2cklog tool is a utility can be used to test the integrity
      of an archive log file and to determine whether or not the log file
                 can be used in the rollforward database command.

      ____________________________________________________________________

________________________________________________________________________________


========================================================
"db2cklog": Processing log file header of "S0000002.LOG"
```

```
"db2cklog": Processing log pages of "S0000002.LOG" (total log pages: "316840")
          ==> page "1" ...
          ==> page "25001" ...
          ==> page "50001" ...
          ==> page "75001" ...
          ==> page "100001" ...
          ==> page "125001" ...
          ==> page "150001" ...
          ==> page "175001" ...
          ==> page "200001" ...
          ==> page "225001" ...
          ==> page "250001" ...
          ==> page "275001" ...
          ==> page "300001" ...

"db2cklog": Finished processing log file "S0000002.LOG". Return code: "0".
========================================================
```

What to do next

If an archive log file fails validation, your response depends on whether or not you have a copy of the log file that can pass validation by the **db2cklog** tool. If you are not sure whether you have a copy of the log file, check the setting for the **logarchmeth2** configuration parameter, which determines whether your database server archives a secondary copy of each log file. If you are validating logs as they are being archive and log mirroring is also configured on your data server, you might still be able to locate a copy of the log file in the log mirror path, as your data server does not recycle log files immediately after archiving.

- If you have a copy of the archive log file, use the **db2cklog** command against that copy. If the copy of the log file passes validation, replace the log file that cannot be read with the valid copy of the log file.
- If you have only one copy of the archive log file and that copy cannot be validated, the log file is beyond repair and cannot be used for rollforward recovery purposes. In this case, you must make a full database backup as soon as possible to establish a new, more recent recovery point that does not depend on the unusable log file for rollforward recovery.

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, issue the **db2dart** command without any parameters. Some options that require parameters, such as the table space ID, are prompted for if they are not explicitly specified on the command line.

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

In a Db2 pureScale environment, some metadata files (such as bufferpool configuration files) exist for each member and are validated or updated on a per-member basis (rather than per-database).

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

Comparison of INSPECT and db2dart:

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

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

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

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

Table 4. Feature comparison of db2dart and INSPECT for table spaces

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

Table 5. Feature comparison of db2dart and INSPECT for data objects

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

Table 6. Feature comparison of db2dart and INSPECT for index objects

Tests performed	db2dart	INSPECT
Check for consistency bit errors	YES	YES
Check the location and length of the index key and whether there is overlapping	YES	YES
Check the ordering of keys in the index	YES	NO
Determine the summary total pages and used pages	NO	YES
Validate contents of internal page header fields	YES	YES
Verify the uniqueness of unique keys	YES	NO
Check for the existence of the data row for a given index entry	NO	YES
Verify each key to a data value	NO	YES
Verify the hash value for key parts with RANDOM ordering	YES	YES (when the INDEXDATA option of INSPECT is used)

Table 7. Feature comparison of db2dart and INSPECT for block map objects

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

Table 8. Feature comparison of db2dart and INSPECT for long field and LOB objects

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

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

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

The **INSPECT** command cannot be used to perform those actions.

Analyzing db2diag log files using db2diag tool

The primary log file intended for use by database and system administrators is the administration notification log. The **db2diag** log files are intended for use by IBM Software Support for troubleshooting purposes.

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

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

Example 1: Filtering the db2diag log files by database name

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

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

Thus you would only see **db2diag** log file 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, base sys utilities, sqleDatabaseQuiesce, probe:2
MESSAGE : ADM7507W  Database quiesce request has completed successfully.
```

Example 2: Filtering the db2diag log files by process ID

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

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

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

```
2006-02-13-14.34.36.027000-300 I18366H421         LEVEL: Severe
PID     : 2200                 TID  : 660         PROC : db2syscs.exe
INSTANCE: DB2                  NODE : 000         DB   : SAMPLE
APPHDL  : 0-1433               APPID: *LOCAL.DB2.060213193043
FUNCTION: Db2, data management, sqldPoolCreate, probe:273
RETCODE : ZRC=0x8002003C=-2147352516=SQLB_BAD_CONTAINER_PATH
          "Bad container path"
```

Example 3: Formatting the db2diag tool output

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

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

An example of the output produced is as follows:

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

For more information, issue the following commands:

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

Example 4: Filtering messages from different facilities

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

- `ALL` which returns records from all facilities
- `MAIN` which returns records from Db2 general diagnostic logs such as the **db2diag** log files and the administration notification log
- `OPTSTATS` which returns records related to optimizer statistics

To read messages from the MAIN facility:

```
db2diag -facility MAIN
```

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

```
db2diag -fac OPTSTATS -level Severe
```

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

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

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

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

Example 5: Merging files and sorting records according to timestamps

This example shows how to merge two or more **db2diag** log files and sort the records according to timestamps.

The two **db2diag** log files to merge are the following ones:

- `db2diag.0.log`; contains records of Level:Error with the following timestamps:
  - 2009-02-26-05.28.49.822637
  - 2009-02-26-05.28.49.835733
  - 2009-02-26-05.28.50.258887
  - 2009-02-26-05.28.50.259685
- `db2diag.1.log`; contains records of Level:Error with the following timestamps:
  - 2009-02-26-05.28.11.480542
  - 2009-02-26-05.28.49.764762
  - 2009-02-26-05.29.11.872184
  - 2009-02-26-05.29.11.872968

To merge the two diagnostic log files and sort the records according to timestamps, execute the following command:

```
db2diag -merge db2diag.0.log db2diag.1.log -fmt %{ts} -level error
```

The result of the merge and sort of the records is as follows:

- 2009-02-26-05.28.11.480542
- 2009-02-26-05.28.49.764762
- 2009-02-26-05.28.49.822637
- 2009-02-26-05.28.49.835733
- 2009-02-26-05.28.50.258887
- 2009-02-26-05.28.50.259685
- 2009-02-26-05.29.11.872184
- 2009-02-26-05.29.11.872968

where the timestamps are merged and sorted chronologically.

Example 6: Merging diagnostic directory path files from a single host and sorting records by timestamps

By default, each member and CF log to a different **db2diag** log file. The following is a list of the three **db2diag** log files to merge:

- `~/sqllib/db2dump/DIAG0000/db2diag.log`

- `~/sqllib/db2dump/DIAG0001/db2diag.log`
- `~/sqllib/db2dump/DIAG0002/db2diag.log`

To merge the three diagnostic log files and sort the records according to timestamps, execute the following command:

```
db2diag -merge
```

Example 7: Merging diagnostic directory path files from multiple hosts and database partitions

This example shows how to obtain an output of all the records from all the diagnostic logs and merge the diagnostic log files from three database partitions on each of two hosts, `bower` and `horton`. The following list shows the six **`db2diag`** log files:

- `~/sqllib/db2dump/HOST_bower/DIAG0000/db2diag.log`
- `~/sqllib/db2dump/HOST_bower/DIAG0001/db2diag.log`
- `~/sqllib/db2dump/HOST_bower/DIAG0002/db2diag.log`
- `~/sqllib/db2dump/HOST_horton/DIAG0003/db2diag.log`
- `~/sqllib/db2dump/HOST_horton/DIAG0004/db2diag.log`
- `~/sqllib/db2dump/HOST_horton/DIAG0005/db2diag.log`

To output the records from all six **`db2diag`** log files, run the following command:

```
db2diag -global
```

To merge all six **`db2diag`** log files in the diagnostic data directory path from all three database partitions on each of the hosts `bower` and `horton` and format the output based on the timestamp, execute the following command:

```
db2diag -global -merge -sdir /temp/keon -fmt %{ts}
```

where `/temp/keon` is a shared directory, shared by the hosts `bower` and `horton`, to store temporary merged files from each host during processing.

Example 8: Filtering and merging only recent diagnostic log entries

In this example, **`db2diag`** log file records are filtered to display only a specific number of recent entries. To display the last 5 formatted records for each of the 3 partitions in a partitioned database environment, merged and formatted by timestamp, enter:

```
db2diag -lastrecords 5 -global -merge -sdir /home/vbmithun -fmt %{ts}

2010-10-08-04.46.02.092192
2010-10-08-04.46.02.092821
2010-10-08-04.46.02.093497
2010-10-08-04.46.02.094431
2010-10-08-04.46.02.095317
2010-10-08-04.46.05.068648
2010-10-08-04.46.05.069212
2010-10-08-04.46.05.069900
2010-10-08-04.46.05.071008
2010-10-08-04.46.05.071831
2010-10-08-04.46.07.302051
2010-10-08-04.46.07.302727
2010-10-08-04.46.07.303544
2010-10-08-04.46.07.304647
2010-10-08-04.46.07.305391
```

You can also filter recent diagnostic log records further to return only messages of a specific level. For example, to return only those records in the last 10 records that have a severe message level, enter:

```
$ db2diag db2diag.log -lastrecords 10 -level Severe -fmt %{ts}

2010-08-11-04.11.33.733807
2010-08-11-04.11.33.735398
```

Example 9: Archiving the db2diag log files

You can use the **db2diag -archive** (or **-A**) option, which is available with IBM Data Server Driver Package and IBM Data Server for ODBC and CLI, to archive the diagnostic log file on an instance-less client. For example:

```
$ db2diag -A
db2diag: Moving "/home/usr1/clidriver/db2dump/db2diag.log"
        to     "/home/usr1/clidriver/db2dump/db2diag.log_2010-09-14-01.16.26"
```

Note: The following commands can produce the same results on an instance-less client.

If you specify options other than **-archive** or **-A**, an error message is returned. For example:

```
$ db2diag -x
db2diag: Unrecognized option: -x

$ db2diag -pid 1234
db2diag: Unrecognized option: -pid
```

Displaying and altering the global registry (UNIX) using db2greg

You can view the global registry using the **db2greg** command on UNIX and Linux platforms.

In Db2 Version 9.7 and higher, the Db2 global profile registries are not recorded in the text file `<DB2DIR>/default.env`. The global registry file `global.reg` is now used to register the Db2 global profile settings related to the current Db2 installation.

The global registry exists only on UNIX and Linux platforms:

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

The global registry consists of three different record types:

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

You can view the global registry with the **db2greg** tool. This tool is located in `sqllib/bin`, and in the `install` directory under `bin` as well (for use when logged in as root).

You can edit the global registry with the **db2greg** tool. Editing the global registry in root installations requires root authority.

Only use the **db2greg** in specific cases as instructed, or when requested to do so by IBM Software Support. Incorrect usage of this command can damage the global registry.

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 website: www.ibm.com/support/docview.wss?rs=71&uid=swg27007053.

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 Software Support for assistance.

For JDBC or SQLJ applications, if you are using the IBM Db2 Driver for SQLJ and JDBC, you can determine the level of the driver by running the db2jcc utility:

```
db2jcc -version

IBM Db2 JDBC Driver Architecture 2.3.63
```

Mimicking databases using db2look

There are many times when it is advantageous to be able to create a database that is similar in structure to another database. For example, rather than testing out new applications or recovery plans on a production system, it makes more sense to create a test system that is similar in structure and data, and to then do the tests against it instead.

This way, the production system will not be affected by the adverse performance impact of the tests or by the accidental destruction of data by an errant application. Also, when you are investigating a problem (such as invalid results, performance issues, and so on), it might be easier to debug the problem on a test system that is identical to the production system.

You can use the **db2look** tool to extract the required DDL statements needed to reproduce the database objects of one database in another database. The tool can also generate the required SQL statements needed to replicate the statistics from the one database to the other, as well as the statements needed to replicate the database configuration, database manager configuration, and registry variables. This is important because the new database might not contain the exact same set of data as the original database but you might still want the same access plans chosen for the two systems. The **db2look** command should only be issued on databases running on Db2 Servers of Version 10.5 and later releases.

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 preceding command. The default database partition groups, buffer pools, and table spaces will not be extracted. This is because they already exist in every database by default. If you want to mimic these, you must alter them yourself manually.

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

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

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

Once you have changed the connect statement, run 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 run 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 contain:

```
...
-- 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 example:

```
-- 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".

Monitoring and troubleshooting using db2pd command

The **db2pd** command is used for troubleshooting because it can return quick and immediate information from the Db2 memory sets.

Overview

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

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

When adding a new node, you can monitor the progress of the operation on the database partition server, that is adding the node, using the **db2pd -addnode** command with the optional `oldviewapps` and `detail` parameters for more detailed information.

If you require a list of event monitors that are currently active or have been, for some reason, deactivated, run the **db2pd -gfw** command. This command also returns statistics and information about the targets, into which event monitors write data, for each fast writer EDU.

Examples

The following list is a collection of examples in which the **db2pd** command can be used to expedite troubleshooting:
- Example 1: Diagnosing a lockwait
- Example 2: Using the **-wlocks** parameter to capture all the locks being waited on
- Example 3: Displaying the table name and schema name of locks
- Example 4: Using the **-apinfo** parameter to capture detailed runtime information about the lock owner and the lock waiter
- Example 5: Using the callout scripts when considering a locking problem
- Example 6: Mapping an application to a dynamic SQL statement
- Example 7: Monitoring memory usage
- Example 8: Determine which application is using up your table space
- Example 9: Monitoring recovery
- Example 10: Determining the amount of resources a transaction is using
- Example 11: Monitoring log usage
- Example 12: Viewing the sysplex list
- Example 13: Generating stack traces
- Example 14: Viewing memory statistics for a database partition
- Example 15: Monitoring the progress of index reorganization
- Example 16: Displaying the top EDUs by processor time consumption and displaying EDU stack information
- Example 17: Displaying agent event metrics
- Example 18: Displaying the extent movement status

The results text show in the examples is an extract of the **db2cmd** command ouput for better readability.

Example 1: Diagnosing a lockwait

If you run **db2pd -db** *databasename* **-locks -transactions -applications -dynamic**, the results are similar to the following ones:

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

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

```
Transactions:
AppHandl [nod-index] TranHdl Locks State Tflag      Tflag2     ...
11       [000-00011] 2       4     READ  0x00000000 0x00000000 ...
12       [000-00012] 3       4     WRITE 0x00000000 0x00000000 ...
```

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

```
Applications:
AppHandl NumAgents CoorPid Status        C-AnchID C-StmtUID L-AnchID L-StmtUID Appid

12       1         1073336 UOW-Waiting   0        0         17       1         ...5602
11       1         1040570 UOW-Executing 17       1         94       1         ...5601
```

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

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

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

Example 2: Using the **-wlocks** parameter to capture all the locks being waited on

If you run **db2pd -wlocks -db pdtest**, results similar to the following ones are generated. They show that the first application (AppHandl 47) is performing an insert on a table and that the second application (AppHandl 46) is performing a select on that table:

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

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

Locks being waited on :
AppHandl TranHdl Lockname                   Type Mode Conv Sts CoorEDU AppName AuthID AppID
47       8       00020004000000000840000652 Row  ..X       G   5160    db2bp   VENUS  ...13730
46       2       00020004000000000840000652 Row  .NS       W   5913    db2bp   VENUS  ...13658
```

Example 3: Displaying the tables name and schema name of locks

Starting in Version 11.1, you can use the `db2pd -locks showlocks` command to display the table name and schema name of locks that are held by applications. You can use this information to diagnose the table and schema that contain the application lock. The table name is displayed in the TableNm column and the schema name is displayed in the SchemaNm column as shown in the following output.

```
Database Member 0 -- Database PDTEST -- Active -- Up 0 days 00:00:10 -- Date 2012-11-06-10.57.18.025767

Locks:
Address            TranHdl    Lockname                   Type         Mode Sts Owner      Dur HoldCount  Att        ReleaseFlg rrIID
0x00002AAAFFFA5F68 3          02000400000020000000000062 MdcBlockLock ..X  G   3          1   0          0x00200000 0x40000000 0
0x00002AAAFFFA7198 3          41414141414A4863ADA1ED24C1 PlanLock     ..S  G   3          1   0          0x00000000 0x40000000 0

TableNm   SchemaNm
T1        YUQZHANG 02000400000020000000000062 SQLP_MDCBLOCK (obj={2;4}, bid=d(0;32;0), x0000200000000000)
N/A       N/A      41414141414A4863ADA1ED24C1 SQLP_PLAN ({41414141 63484A41 24EDA1AD}, loading=0)
```

You can also use the db2pd -wlocks detail command to display the table name, schema name, and application node of locks that are being waited on as shown in the following output.

```
Database Member 0 -- Database PDTEST -- Active -- Up 0 days 00:00:35 -- Date 2012-11-06-11.11.32.403994

Locks being waited on :
AppHandl [nod-index] TranHdl    Lockname                   Type       Mode Conv Sts CoorEDU  AppName AuthID   AppID
19       [000-00019] 3          02000400000000000000000054  TableLock  ..X       G   18       db2bp   YUQZHANG *LOCAL.yuqzhang.121106161112
21       [000-00021] 15         02000400000000000000000054  TableLock  .IS       W   45       db2bp   YUQZHANG *LOCAL.yuqzhang.121106161114

TableNm  SchemaNm AppNode
PDTEST   YUQZHANG hotel71
PDTEST   YUQZHANG hotel71
```

Example 4: Using the **-apinfo** parameter to capture detailed runtime information about the lock owner and the lock waiter

The following sample output was generated under the same conditions as those for Example 2:

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

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

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

  ClientUserID :           n/a
  ClientWrkstnName :       n/a
  ClientApplName :         n/a
  ClientAccntng :          n/a

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


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

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

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

  ClientUserID :           n/a
  ClientWrkstnName :       n/a
  ClientApplName :         n/a
  ClientAccntng :          n/a

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

Example 5: Using the callout scripts when considering a locking problem

To use the callout scripts, 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 script file. An example of the output provided when the db2cos script file contains a **db2pd -db sample -locks** command is as follows:

```
Lock Timeout Caught
Thu Feb 17 01:40:04 EST 2006
Instance DB2
Database: SAMPLE
Partition Number: 0
```

```
PID: 940
TID: 2136
Function: sqlplnfd
Component: lock manager
Probe: 999
Timestamp: 2006-02-17-01.40.04.106000
AppID: *LOCAL.DB2...
AppHdl:
...
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:06:53
Locks:
Address    TranHdl Lockname                   Type Mode Sts Owner Dur HldCnt Att Rlse
0x402C6B30 3       00020003000000040000000052 Row  ..X  W*  3     1   0      0   0x40
```

In the output, `W*` indicates the lock that experienced the timeout. In this case, a lockwait has occurred. A lock timeout can also occur when a lock is being converted to a higher mode. This is indicated by `C*` in the output.

You can map the results to a transaction, an application, an agent, or even an SQL statement with the output provided by other **db2pd** commands in the db2cos file. You can narrow down the output or use other commands to collect the information that you need. For example, you can use the **db2pd -locks wait** parameters to print only locks with a wait status. You can also use the **-app** and **-agent** parameters.

Example 6: Mapping an application to a dynamic SQL statement

The command **db2pd -applications -dynamic** reports the current and last anchor ID and statement unique ID for dynamic SQL statements. This allows direct mapping from an application to a dynamic SQL statement.

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

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

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

Example 7: Monitoring memory usage

The **db2pd -memblock** command can be useful when you are trying to understand memory usage, as shown in the following sample output:

```
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 memory 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 operating systems. For example, if you run **db2pd -memb pid=159770**, results similar to the following ones are generated:

```
All memory blocks in Private set.

PoolID     PoolName     BlockAge    Size(Bytes) I LOC   File
88         private      1           2488        1 172   4283993058
88         private      2           1608        1 172   4283993058
88         private      3           4928        1 172   4283993058
88         private      4           7336        1 172   4283993058
88         private      5           32          1 172   4283993058
88         private      6           6728        1 172   4283993058
88         private      7           168         1 172   4283993058
88         private      8           24          1 172   4283993058
88         private      9           408         1 172   4283993058
88         private      10          1072        1 172   4283993058
88         private      11          3464        1 172   4283993058
88         private      12          80          1 172   4283993058
```

```
88        private      13        480        1 1534  862348285
88        private      14        480        1 1939  862348285
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
```

Example 8: Determine which application is using up your table space

Using **db2pd -tcbstats** command, you can identify the number of inserts for a table. The following example shows sample information for a user-defined global temporary table called TEMP1:

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

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

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

```
Tablespace 3 Configuration:
Type Content PageSz ExtentSz Auto Prefetch BufID FSC RSE NumCntrs MaxStripe LastConsecPg Name
DMS  UsrTmp  4096   32       Yes  32       1     On  Yes 1        0         31           TEMPSPACE2
Tablespace 3 Statistics:
TotalPgs UsablePgs UsedPgs PndFreePgs FreePgs HWM  State      MinRecTime NQuiescers
5000     4960      1088    0          3872    1088 0x00000000 0          0

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

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

The MinRecTime column returns a value that is a UNIX timestamp in a UTC timezone format. To convert the value to a GMT time zone format you can use the Db2 timestamp function. For example, if MinRecTime returns a value of 1369626329, to convert this value to a GMT format run the following statement:

```
db2 "values timestamp('1970-01-01-00.00.00') + 1369626329 seconds"
```

The query will return a GMT value of 2013-05-27-03.45.29.000000.

You can see if the reclaimable space feature is enabled in the Reclaimable Space Enabled (RSE) column. The FreePgs column shows that space is filling up. As the free pages value decreases, there is less space available. Notice also that the value for FreePgs plus the value for UsedPgs equals the value of UsablePgs.

Once this is known, you can identify the dynamic SQL statement that is using the table TEMP1 by running the **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:
AnchID StmtUID NumEnv NumVar NumRef NumExe Text
78        1       2     2      3      2     declare global temporary table temp1 ...
253    1       1      1      24     24     insert into session.temp1 values('TEST')
```

Finally, you can map the information from the preceding output to the applications output to identify the application by running **db2pd -db sample -app**.

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

L-AnchID L-StmtUID  Appid
253      1          *LOCAL.db2inst1.050202160426
```

You can use the anchor ID (AnchID) value that identified the dynamic SQL statement to identify the associated application. The results show that the last anchor ID (L-AnchID) value is the same as the anchor ID (AnchID) value. You use the results from one run of **db2pd** in the next run of **db2pd**.

The output from **db2pd -agent** shows the number of rows read (in the Rowsread column) and rows written (in the Rowswrtn column) by the application. These values give you an idea of what the application has completed and what the application still has to complete, as shown in the following sample output:

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

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

You can map the values for AppHandl and AgentPid resulting from running the **db2pd -agent** command to the corresponding values for AppHandl and CoorPiid resulting from running the **db2pd -app** command.

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

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

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

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

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

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

Tablespace Statistics:
Id TotalPgs   UsablePgs  UsedPgs    PndFreePgs FreePgs    HWM    State      MinRecTime NQuiescers
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 application handles 30 and 31 (because you saw them in the **-tcbstats** output) by using the command **db2pd -app**:

```
Applications:
AppHandl NumAgents CoorPid Status        C-AnchID C-StmtUID L-AnchID L-StmtUID Appid
31       1         4784182 UOW-Waiting   0        0         107      1         ...4142
30       1         8966270 UOW-Executing 107      1         107      1         ...4013
```

Finally, map the information from the preceding output to the Dynamic SQL output obtained by running the **db2pd -dyn** command:

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

Example 9: Monitoring recovery

If you run the command **db2pd -recovery**, the output shows several counters that you can use to verify that recovery is progressing, as shown in the following sample output. The Current Log and Current LSO values provide the log position. The CompletedWork value is the number of bytes completed thus far.

```
Recovery:
Recovery Status     0x00000401
Current Log         S0000005.LOG
Current LSN         0000001F07BC
Current LSO         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
```

Example 10: Determining the amount of resources a transaction is using

If you run the command **db2pd -transactions**, the output shows the number of locks, the first log sequence number (LSN), the last LSN, the first LSO, the last LSO, the log space that is used, and the space reserved. The output also displays the total number of application commits and the total number of application rollbacks. Knowing the total number of application commits and rollbacks can be useful for understanding the behavior of a transaction. The following is a sample output of the db2pd -transactions command.

```
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        Firstlso        Lastlso
0x00000000 0x0000001A4212 0x0000001C2022 0x000001072262  0x0000010B2C8C
0x00000000 0x000000107320 0x0000001S3462 0x000001057574  0x0000010B3340
0x00000000 0x0000001BC00C 0x0000001X2F03 0x00000107CF0C  0x0000010B2FDE
LogSpace   SpaceReserved  TID            AxRegCnt   GXID
4518       95450          0x000000000451 1          0
6576       139670         0x0000000003E0 1          0
3762       79266          0x000000000472 1          0

Total Application commits   : 23
Total Application rollbacks : 39
```

Example 11: Monitoring log usage

The command **db2pd -logs** is useful for monitoring log usage for a database. By using thePages Written value, as shown in the following sample output, you can determine whether the log usage is increasing:

```
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         StartLSO       State      Size  Pages  Filename
0x000000023001BF58  0x00000022F032   0x000001B58000 0x00000000 1000  1000   S0000002.LOG
0x000000023001BE98  0x000000000000   0x000001F40000 0x00000000 1000  1000   S0000003.LOG
0x0000000230008F58  0x000000000000   0x000002328000 0x00000000 1000  1000   S0000004.LOG
```

You can identify two types of problems by using this output:

- If the most recent log archive fails, Archive Status is set to a value of Failure. If there is an ongoing archive failure, preventing logs from being archived at all, Archive Status is set to a value of First Failure.
- If log archiving is proceeding very slowly, the Next Log to Archive value is lower than the Current Log Number value. If archiving is very slow, space for active logs might run out, which in turn might prevent any data changes from occurring in the database.

Note: S0000003.LOG and S0000004.LOG do not contain any log records yet and therefore the StartLSN is 0x0

Example 12: Viewing the sysplex list

Without the **db2pd -sysplex** command showing the following sample output, the only other way to report the sysplex list is by using 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
```

Example 13: Generating stack traces

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

You can obtain the current call stack for a particular engine dispatchable unit (EDU) by issuing the command **db2pd -stack** *eduid*, as shown in the following example:

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

If the call stacks for all of the Db2 processes are desired, use the command **db2pd -stack all**, for example (on Windows operating systems):

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

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

You can also redirect the output of the **db2pdb -stacks** command for **db2sysc** processes to a specific directory path with the **dumpdir** parameter. The output can be redirected for a specific duration only with the **timeout** parameter. For example, to redirect the output of stack traces for all EDUs in **db2sysc** processes to /home/waleed/mydir for 30 seconds, issue the following command:

```
db2pd -alldbp -stack all dumpdir=/home/waleed/mydir timeout=30
```

Example 14: Viewing instance memory statistics for a database partition

The **db2pd -dbptnmem** command shows how much memory the Db2 server is currently consuming and, at a high level, which areas of the server are using that memory. Instance memory usage includes not only actual system memory consumption/commitment but also configured allowances that may not be in use/committed. Therefore, memory usage statistics for the Db2 instance are not directly comparable with memory usage statistics reported by operating system monitoring tools.

The following example shows the output from running **db2pd -dbptnmem** on an AIX machine:

```
Database Partition Memory Controller Statistics

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

The descriptions of these data fields and columns are as follows:

Controller Automatic
: Indicates the memory controller setting. When set to "Y" (**instance_memory** configuration parameter is set to AUTOMATIC), the calculated limit is only enforced when the product license contains a memory limit. When set to "N" the stated limit is enforced.

Memory Limit
: If an instance memory limit is enforced, the value of the **instance_memory** configuration parameter is the upper bound limit of instance memory that can be consumed.

Current usage
: The amount of instance memory the server is currently consuming.

HWM usage
: The high water mark (HWM) or peak instance memory usage that has been consumed since the activation of the database partition (when the **db2start** command was run).

Cached memory
: The amount of the current usage that may be reclaimed . This applies when instance memory usage is approaching an enforced limit, and cached usage by one or more consumers may need to be reduced to allow some other consumer to grow.

Following is the continuation of the sample output from running **db2pd -dbptnmem** on an AIX operating system:

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

All registered “consumers” of instance memory within the Db2 server are listed with the amount of the total instance memory they are consuming. The column descriptions are as follows:

Name A short, distinguishing name of a consumer of instance memory, such as the following ones:

APPL-*dbname*
: Application memory consumed for database *dbname*

DBMS-*name*
: Global database manager memory requirements

FMP_RESOURCES
: Memory required to communicate with **db2fmps**

PRIVATE
: Miscellaneous private memory requirements

FCM_RESOURCES
: Fast Communication Manager resources

LCL-*pid*
: The memory segment used to communicate with local applications

DB-*dbname*
: Database memory consumed for database *dbname*

Mem Used (KB)
: The amount of instance memory that is currently allotted to the consumer

HWM Used (KB)
The high-water mark (HWM) or the peak instance memory, that the consumer has used

Cached (KB)
Of the Mem Used (KB), the amount of instance memory that may be reclaimed for this consumer.

Example 15: Monitoring the progress of index reorganization

In Db2 Version 9.8 Fix Pack 3 and later fix packs, the progress report of an index reorganization has the following characteristics:

- The **db2pd -reorgs index** command reports index reorg progress for partitioned indexes (Fix Pack 1 introduced support for only non-partitioned indexes).
- The **db2pd -reorgs index** command supports the monitoring of index reorg at the partition level (that is, during reorganization of a single partition).
- The reorg progress for non-partitioned and partitioned indexes is reported in separate outputs. One output shows the reorg progress for non-partitioned indexes, and the following outputs show the reorg progress for partitioned indexes on each table partition; the index reorg statistics of only one partition is reported in each output.
- Non-partitioned indexes are processed first, followed by partitioned indexes in serial fashion.
- The **db2pd -reorgs index** command displays the following additional information fields in the output for partitioned indexes:
  - `MaxPartition` - Total number of partitions for the table being processed. For partition-level reorg, `MaxPartition` will always have a value of 1 since only a single partition is being reorganized.
  - `PartitionID` - The data partition identifier for the partition being processed.

The following example shows an output obtained using the **db2pd -reorgs index** command which reports the index reorg progress for a range-partitioned table with 2 partitions.

Note: The first output reports the Index Reorg Stats of the non-partitioned indexes. The following outputs report the Index Reorg Stats of the partitioned indexes on each partition.

```
Index Reorg Stats:
Retrieval Time: 02/08/2010 23:04:21
TbspaceID: -6       TableID: -32768
Schema: ZORAN    TableName: BIGRPT
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:03:55   End Time: 02/08/2010 23:04:04
Total Duration: 00:00:08
Prev Index Duration:  -
Cur Index Start: -
Cur Index: 0           Max Index: 2           Index ID: 0
Cur Phase: 0          ( -     )  Max Phase: 0
Cur Count: 0                     Max Count: 0
Total Row Count: 750000

Retrieval Time: 02/08/2010 23:04:21
TbspaceID: 2        TableID: 5
Schema: ZORAN    TableName: BIGRPT
PartitionID: 0      MaxPartition: 2
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:04:04   End Time: 02/08/2010 23:04:08
```

```
Total Duration: 00:00:04
Prev Index Duration:  -
Cur Index Start: -
Cur Index: 0           Max Index: 2           Index ID: 0
Cur Phase: 0          ( -     )   Max Phase: 0
Cur Count: 0                      Max Count: 0
Total Row Count: 375000

Retrieval Time: 02/08/2010 23:04:21
TbspaceID: 2       TableID: 6
Schema: ZORAN    TableName: BIGRPT
PartitionID: 1      MaxPartition: 2
Access: Allow none
Status: Completed
Start Time: 02/08/2010 23:04:08   End Time: 02/08/2010 23:04:12
Total Duration: 00:00:04
Prev Index Duration:  -
Cur Index Start: -
Cur Index: 0           Max Index: 2           Index ID: 0
Cur Phase: 0          ( -     )   Max Phase: 0
Cur Count: 0                      Max Count: 0
Total Row Count: 375000
```

Example 16: Displaying the top EDUs by processor time consumption and displaying EDU stack information

If you issue the **db2pd** command with the **-edus** parameter option, the output lists all engine dispatchable units (EDUs). Output for EDUs can be returned at the level of granularity you specify, such as at the instance level or at the member. On Linux and UNIX operating systems only, you can also specify the **interval** parameter suboption so that two snapshots of all EDUs are taken, separated by an interval you specify. When the **interval** parameter is specified, two additional columns in the output indicate the delta of processor user time (USR DELTA column) and the delta of processor system time (SYS DELTA column) across the interval.

In the following example, the deltas for processor user time and processor system time are given across a five-second interval:

```
$ db2pd -edus interval=5

Database Partition 0 -- Active -- Up 0 days 00:53:29 -- Date 06/04/2010 03:34:59

List of all EDUs for database partition 0

db2sysc PID: 1249522
db2wdog PID: 2068678

EDU ID TID  Kernel TID EDU Name                        USR       SYS       USR DELTA  SYS DELTA
===============================================================================================
6957   6957 13889683   db2agntdp (SAMPLE  ) 0          58.238506  0.820466  1.160726   0.014721
6700   6700 11542589   db2agent (SAMPLE) 0             52.856696  0.754420  1.114821   0.015007
5675   5675 4559055    db2agntdp (SAMPLE  ) 0          60.386779  0.854234  0.609233   0.014304
3088   3088 13951225   db2agntdp (SAMPLE  ) 0          80.073489  2.249843  0.499766   0.006247
3615   3615 2887875    db2loggw (SAMPLE) 0             0.939891  0.410493  0.011694   0.004204
4900   4900 6344925    db2pfchr (SAMPLE) 0             1.748413  0.014378  0.014343   0.000103
7986   7986 13701145   db2agntdp (SAMPLE  ) 0          1.410225  0.025900  0.003636   0.000074
2571   2571 8503329    db2ipccm 0                      0.251349  0.083787  0.002551   0.000857
7729   7729 14168193   db2agntdp (SAMPLE  ) 0          1.717323  0.029477  0.000998   0.000038
7472   7472 11853991   db2agnta (SAMPLE) 0             1.860115  0.032926  0.000860   0.000012
3358   3358 2347127    db2loggr (SAMPLE) 0             0.151042  0.184726  0.000387   0.000458
515    515  13820091   db2aiothr 0                     0.405538  0.312007  0.000189   0.000178
7215   7215 2539753    db2agntdp (SAMPLE  ) 0          1.165350  0.019466  0.000291   0.000008
6185   6185 2322517    db2wlmd (SAMPLE) 0              0.061674  0.034093  0.000169   0.000100
6442   6442 2756793    db2evmli (DB2DETAILDEADLOCK) 0  0.072142  0.052436  0.000092   0.000063
4129   4129 15900799   db2glock (SAMPLE) 0             0.013239  0.000741  0.000064   0.000001
2      2    11739383   db2alarm 0                      0.036904  0.028367  0.000009   0.000009
4386   4386 13361367   db2dlock (SAMPLE) 0             0.015653  0.001281  0.000014   0.000003
1029   1029 15040579   db2fcms 0                       0.041929  0.016598  0.000010   0.000004
5414   5414 14471309   db2pfchr (SAMPLE) 0             0.000093  0.000002  0.000000   0.000000
258    258  13656311   db2sysc 0                       8.369967  0.263539  0.000000   0.000000
5157   5157 7934145    db2pfchr (SAMPLE) 0             0.027598  0.000177  0.000000   0.000000
```

```
1543   1543 2670647    db2fcmr 0                          0.004191   0.000079  0.000000   0.000000
1286   1286 8417339    db2extev 0                         0.000312   0.000043  0.000000   0.000000
2314   2314 14360813   db2licc 0                          0.000371   0.000051  0.000000   0.000000
5928   5928 3137537    db2taskd (SAMPLE) 0                0.004903   0.000572  0.000000   0.000000
3872   3872 2310357    db2lfr (SAMPLE) 0                  0.000126   0.000007  0.000000   0.000000
4643   4643 11694287   db2pclnr (SAMPLE) 0                0.000094   0.000002  0.000000   0.000000
1800   1800 5800175    db2extev 0                         0.001212   0.002137  0.000000   0.000000
772    772  7925817    db2thcln 0                         0.000429   0.000072  0.000000   0.000000
2057   2057 6868993    db2pdbc 0                          0.002423   0.001603  0.000000   0.000000
2828   2828 10866809   db2resync 0                        0.016764   0.003098  0.000000   0.000000
```

To provide information only about the EDUs that are the top consumers of processor time and to reduce the amount of output returned, you can further include the **top** parameter option. In the following example, only the top five EDUs are returned, across an interval of 5 seconds. Stack information is also returned, and can be found stored separately in the directory path specified by DUMPDIR, which defaults to **diagpath**.

```
$ db2pd -edus interval=5 top=5 stacks

Database Partition 0 -- Active -- Up 0 days 00:54:00 -- Date 06/04/2010 03:35:30

List of all EDUs for database partition 0

db2sysc PID: 1249522
db2wdog PID: 2068678

EDU ID TID   Kernel TID EDU Name            USR       SYS       USR DELTA  SYS DELTA
=====================================================================================
3358   3358  2347127    db2loggr (SAMPLE) 0  0.154906  0.189223  0.001087   0.001363
3615   3615  2887875    db2loggw (SAMPLE) 0  0.962744  0.419617  0.001779   0.000481
515    515   13820091   db2aiothr 0          0.408039  0.314045  0.000658   0.000543
258    258   13656311   db2sysc 0            8.371388  0.264812  0.000653   0.000474
6700   6700  11542589   db2agent (SAMPLE) 0  54.814420 0.783323  0.000455   0.000310


$ ls -ltr
total 552
drwxrwxr-t    2 vbmithun build        256 05-31 09:59 events/
drwxrwxr-t    2 vbmithun build        256 06-04 03:17 stmmlog/
-rw-r--r--    1 vbmithun build      46413 06-04 03:35 1249522.3358.000.stack.txt
-rw-r--r--    1 vbmithun build      22819 06-04 03:35 1249522.3615.000.stack.txt
-rw-r--r--    1 vbmithun build      20387 06-04 03:35 1249522.515.000.stack.txt
-rw-r--r--    1 vbmithun build      50426 06-04 03:35 1249522.258.000.stack.txt
-rw-r--r--    1 vbmithun build     314596 06-04 03:35 1249522.6700.000.stack.txt
-rw-r--r--    1 vbmithun build      94913 06-04 03:35 1249522.000.processObj.txt
```

Example 17: Displaying agent event metrics

The **db2pd** command supports returning event metrics for agents. If you need to determine whether an agent changed state during a specific period of time, use the event option together with the **-agents** parameter. The AGENT_STATE_LAST_UPDATE_TIME(Tick Value) column that is returned shows the last time that the event being processed by the agent was changed. Together with a previously obtained value for AGENT_STATE_LAST_UPDATE_TIME(Tick Value), you can determine whether an agent has moved on to a new task or whether it continues to process the same task over an extended period of time.

```
db2pd -agents event
Database Partition 0 -- Active -- Up 0 days 03:18:52 -- Date 06/27/2011 11:47:10

Agents:
Current agents:      12
Idle agents:         0
Active coord agents: 10
Active agents total: 10
Pooled coord agents: 2
Pooled agents total: 2
```

```
AGENT_STATE_LAST_UPDATE_TIME(Tick Value) EVENT_STATE EVENT_TYPE EVENT_OBJECT EVENT_OBJECT_NAME
2011-06-27-14.44.38.859785(...968075)    IDLE        WAIT       REQUEST      n/a
```

Example 18: Displaying the extent movement

You can display the extent movement status of your table space by issuing the `db2pd -extentmovement -db` *`dbName`* command.

```
$ db2pd -extentmovement -db PDTEST

Database Member 0 -- Database PDTEST -- Active -- Up 0 days 00:04:33 -- Date 2012-10-26-11.19.52.056414

Extent Movement:
Address            TbspName Current Last Moved Left TotalTime
0x00002AAB356D4BA0 DAVID    1168    1169 33    426  329636
```

Collecting environment information with the db2support command

When it comes to collecting information for a Db2 problem, the most important Db2 utility you need to run is **db2support**. The **db2support** command automatically collects 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.

About this task

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.

Procedure

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

  To collect all necessary information without an error, run the **db2support** utility as a user with SYSADM authority, such as an instance owner. If you run the command without SYSADM authority, SQL errors (for example, SQL1092N) might result when the utility runs commands such as **QUERY CLIENT** or **LIST ACTIVE DATABASES**.

  To use the **db2support** command to help collect information for IBM Software Support, run the **db2support** command while the system is experiencing the problem. That way, the tool collects timely information, such as operating system performance details. If you cannot run the utility at the time of the problem, you can still issue the **db2support** command after the problem stops because some first occurrence data capture (FODC) diagnostic files are produced automatically.

  If an FODC package is stored in a directory path that is different from the default diagnostic path or not in a path specified by an FODCPATH setting, indicate the FODC path to the **db2support** command with the **-fodcpath** parameter, so that the FODC package can be included in the `db2support.zip` file.

  The following basic invocation is typically sufficient for collecting most of the information required to debug a problem, unless you need to include the path to an FODC package with the **-fodcpath** parameter:

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

  The **db2support** tool collects most diagnostic data specific to Db2 pureScale components by default. If you specify the **-purescale**, **-cm**, **-cfs**, or **-udapl** parameter, the **db2support** command collects additional diagnostic data that is space intensive or takes a longer time to collect, but helps determining the source of problem faster in Db2 pureScale environments.

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.

Results

The type of information that **db2support** captures depends on the way the command is invoked, whether the database manager is started, and whether it is possible to connect to the database.

The **db2support** utility collects the following information under all conditions:

- **db2diag** log files
- All trap files
- Locklist files
- Dump files
- Various system-related files
- Output from various system commands
- `db2cli.ini`
- `db2dsdriver.cfg`

Depending on the circumstances, the **db2support** utility might also collect:

- Active log files
- Buffer pool and table space (`SQLSPCS.1` and `SQLSPCS.2`) control files (with **-d** option)
- Contents of the `db2dump` directory
- Extended system information (with **-s** option)
- Database configuration settings (with **-d** option)
- Database manager configuration settings files
- First occurrence data capture (FODC) information (with the **-fodc** and **-fodcpath** options)
- Log file header file (with **-d** option)
- Recovery history file (with **-d** option)
- Formatted data for SYSIBM.SYSTABLES, SYSIBM.SYSINDEXES, and SYSIBM.SYSDATAPARTITIONS system catalog tables (with **-d** option, the database not activated and the **db2support** command not in optimizer mode)
- Workload manager (WLM) related information for optimizer mode (with **-wlm** option)
- Specified system user information (with **-su** option)
- Specified system group information (with **-sg** option)

The **db2support** utility collects the following information in Db2 pureScale environments:

- Diagnostic data for Db2 pureScale components, such as cluster manager, cluster file system, and uDAPL
- Additional diagnostic data for cluster manager (with **-cm** option)
- Additional diagnostic data for cluster file system (with **-cfs** option)
- Additional diagnostic data for uDAPL (with **-udapl** option)
- Additional Db2 diagnostic data (with **-purescale** option)

The HTML report `db2support.html` always includes 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 installation path information
- Contents of `db2nodes.cfg`
- Number of processors and disks and how much memory
- List of databases in the instance
- Registry information and environment, including PATH and LIBPATH
- Disk free space for current file system and inodes for UNIX
- Java™ SDK level
- Java JCC version
- Java JCC configuration
- Database manager configuration
- **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
- The result of the **db2prereqcheck** command
- List of all installed software
- Db2 license information
- Db2 compliance report
- Audit configuration information
- CLI configuration information
- Problem determination settings
- Status of the **db2trc** command
- Listing of log directory

The `db2support.html` file contains the following additional information if the Db2 database manager is started:
- Client connection state
- Database and database manager configuration (Database configuration requires the **-d** option)
- Application snapshot
- 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 when a connection to the database was 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
- Db2 workload manager information
- Listing of the database recovery history file
- Optimizer database configuration
- Database configuration
- Nodegroup information
- Storage group information
- Number of string IDs
- List of tables

A `db2support.html` file is included at the top level of the **db2support** package to help you to quickly search for any diagnostic data that is collected by the **db2support** command. This HTML file includes links of the data collected in the `db2support.html` file that point to its corresponding flat files in the subdirectory of the `db2support` package. A plain text version of the map file called `db2support.map` file is also included in the **db2support** package.

Example contents of db2support.zip file

You can use the **db2support** command with the **-unzip** parameter to extract the contents of the `db2support.zip` file locally, optionally specifying the directory path where you want the contents extracted to. You can also use the **-unzip** option to extract the contents of archived diagnostic data without the need for additional software. If you want to know only what files are included in a `db2support.zip` file, without extracting the actual content, you can instead use the **-unzip list** parameters with the **db2support** command.

For an example of the contents of a `db2support.zip` file, the following command was executed:

```
db2support . -d sample -c -f -st "select * from staff"
```

Extracting the `db2support.zip` file, the following files and directories were collected:

- DB2CONFIG/ - Configuration information (for example, database, database manager, BP, CLI, and Java developer kit, among others)
- DB2DUMP/ - db2diag log file contents for the past three days
- DB2MISC/ - List of the `sqllib` directory

- DB2SNAP/ - Output of Db2 commands (for example,**db2set**, **LIST TABLES**, **LIST INDOUBT TRANSACTIONS**, and **LIST APPLICATIONS**)
- PURESCALE/- Diagnostic information for Db2 pureScale components, such as cluster manager, cluster file system and uDAPL
- db2supp_opt.zip - Diagnostic information for optimizer problems
- db2supp_system.zip - Operating system information
- db2support.html - Map to flat files collected in each subdirectory of the `db2support.zip` file listed in HTML format and diagnostic information formatted into HTML sections
- db2support.log - Diagnostic log information for **db2support** collection
- db2support_options.in - Command-line options used to start the **db2support** collection
- db2support.map - Map to flat files collected in each subdirectory of the `db2support.zip` file listed in plain text format

Information about Optimizer can be found in the `db2supp_opt.zip` file. Extraction of this file finds the following directories:
- OPTIMIZER/ - Diagnostic information for optimizer problems
- OPTIMIZER/optimizer.log - File contains a log of all activities. If db2support prompts messages, for example, DBT7116E, you might find more information within this log file.
- OPTIMIZER/CATALOGS - All the catalogs with LOBs in the following subdirectories (generated only if the LOB column in the catalog table is not empty):
  - FUNCTIONS
  - INDEXES
  - NODEGROUPS
  - ROUTINES
  - SEQUENCES
  - TABLES
  - VIEWS
- OPTIMIZER/DB2DUMP - db2serv output (`serv.*` and `serv2.*` output files)

System information can be found in the `db2supp_system.zip` file. Extraction of this file finds the following file and directories:
- DB2CONFIG/ - db2cli.ini (files from `~/sqllib/cfg`)
- DB2MISC/ - DB2SYSTM file (binary), among others
- OSCONFIG/ - Different operating system information files (for example, `netstat`, `services`, `vfs`, `ulimit`, and `hosts`)
- OSSNAP/ - Operating system snapshots (for example, **iostat**, **netstat**, **uptime**, **vmstat**, and **ps_elf**)
- SQLDBDIR/ - Important buffer pool meta files (`~/sqllib/sqldbdir`)
- SQLGWDIR/ - DCS directory (files from `~/sqllib/sqlgwdir`)
- SQLNODIR/ - Node directory (files from `~/sqllib/sqlnodir`)
- SPMLOG/ - Files from `~/sqllib/spmlog`
- report.log - Log of all collection activities

Scripts for Troubleshooting

The following scripts are available to help you troubleshoot.

Troubleshooting Db2 application hangs:

You can use the **`db2_hang_analyze`** script to detect and analyze possible application hangs in a Db2 database. It is a Perl script located in the `sqllib/samples/pd/` directory.

About this task

If an application is not responding or is not finishing as quickly as you expect, the application might be hanging. For example, you might try to run a query against the database, but the query does not respond. You might suspect that a hang occurred. However, the query might be taking a long time to run, as is the case with queries that involve Cartesian products on large tables.

Root user authority required.

Procedure

To troubleshoot a possible application hang, issue the **`db2_hang_analyze`** script. An example follows:

```
$HOME/sqllib/samples/pd/db2_hang_analyze -db dbname -log
```

Unless you terminate the script by pressing Ctrl-C or Ctrl-Z, the script runs until it detects a possible hang. If you specify the **`-log`** parameter, notifications, warnings, and errors are printed to a log file. If a hang is detected, a list of hanging applications is written to a report file. An example of the report file follows:

```
cat /home/hotel32/shenli/sqllib/db2dump/db2_hang_analyze.20130125.14.41.38.10297.report
APPLICATION HANG DETECTION: Started on Fri Jan 25 14:41:38 EST 2013
Sleeptime               :  60 seconds
Timer Limit             :  300 seconds
Node Member             :  default
Retry Limit             :  3
Log                     :  yes
SQL                     :  no
Event Metrics Available :  yes
Logfile                 :  db2_hang_analyze.20130125.14.41.38.10297.log
Script PID              :  10297
CPU Threshold           :  0.1%
Post Detection Script   :  none
Path                    :  /home/hotel32/shenli/sqllib/db2dump/

POTENTIAL APPLICATIONS HANGING: 3 application(s).
Apphdl             :  7
Status             :  CommitActive
AgentEDUID         :  16

Apphdl             :  9
Status             :  CommitActive
AgentEDUID         :  28

Apphdl             :  19
Status             :  CommitActive
AgentEDUID         :  37


APPLICATION HANG DETECTION: Ended on Fri Jan 25 14:46:47 EST 2013
```

The report file shows that three applications are hanging.

What to do next

Using the report from the **db2_hang_analyze** script, you can make an informed decision on the best course of action to take with the hanging applications. For example, you can retrieve a stacktrace of the hanging application three times at 60-second intervals, by issuing the following command:

```
db2pd -stack eduid -repeat 60 3
```

where *eduid* is the AgentEDUID number from the report file produced when you run the **db2_hang_analyze** script.

db2_hang_analyze - Detect and analyze possible application hangs script:

Detects and analyzes possible Db2 database application hangs by using various metrics that are gathered from the **db2pd** command. The **db2_hang_analyze** script is available only on Linux and UNIX operating systems.

The **db2_hang_analyze** script is a Perl script that runs indefinitely. It gathers metrics on each application for each iteration, and checks if the application is active over certain time interval (the default value is 300 seconds). If the application is not active over the time interval, it is flagged as hanging. If a potential hang is detected, a list of applications is written to a report file. You can terminate the script by pressing Ctrl-C or Ctrl-Z.

The **db2_hang_analyze** script is in the `sqllib/samples/pd/` directory.

Authorization

You require one of the following authorities:
- SYSADM
- SYSCTRL
- SYSMAINT
- SYSMON

Required connection

Active database

Syntax

```
►►─db2_hang_analyze─db─dbname─┬───────────────────────────┬────────────►◄
                              ├─member─member-number──────┤
                              ├─timerlimit─seconds────────┤
                              ├─Sleeptime─Seconds─────────┤
                              ├─retrylimit─attempts───────┤
                              ├─path─directory────────────┤
                              ├─cputhreshold─percentage───┤
                              ├─exec─script_path──────────┤
                              ├─log───────────────────────┤
                              ├─sql───────────────────────┤
                              ├─list──────────────────────┤
                              └─h─────────────────────────┘
```

Parameters

-db *dbname*
: Specifies the database for which to detect hangs. This parameter is required.

member *member number*
: Specifies the member on which the script is issued. If this option is not specified, the script is issued on the current member. If the instance is in serial mode, this parameter is ignored.

-timerlimit *seconds*
: Specifies the amount of time, in seconds, that an application is idle (no change in metrics) before it is determined to be hanging. The default value is 300 seconds.

-sleeptime *seconds*
: Specifies the amount of time to wait, in seconds, before the script starts the next iteration of hang analysis. The default value is 60 seconds.

-retrylimit *attempts*
: Specifies the number of times a **db2pd** command is run after it fails or timeouts. The default value is three attempts.

-path *directory*
: Specifies the full directory path where the report file and log file are stored. The default value is specified by the `DIAGPATH` database manager configuration parameters.

-cputhreshold *percentage*
: Specifies the threshold for a change in the percentage of processor time that an application uses. If an application's consumption change exceeds the threshold, the application is flagged as active. The default value is 0.1 (0.1%).

-exec *script_path*
: Specifies a path to a script that runs after a hang is detected. The script is likely customized to further troubleshooting of the hang. The output of the script is placed in the `DIAGPATH` directory in the format of `db2_hang_analyze.<`*`timestamp`*`>.exec`. The default value is off, meaning that no script is run after a hang is detected.

-log
: Specify `yes` to print notification about possible hanging applications, warnings, and errors to the log file. The default value is `no`.

-sql
: Specify `yes` to print the most recent SQL statement that was issued by the hanging application, if the data exists. The default value is `no`.

-list
: Generates a list of different **db2_hang_analyze** scripts that are running across your system. In this list, you can specify which **db2_hang_analyze** scripts to terminate.

-h
: Displays help information.

Example 1

In the following example, thescript monitors the SAMPLE database to detect any possible application hangs. Various metrics from the **db2pd** command are collected on every application every 60 seconds. If an application is determined to be hanging, the script writes a report and then exits.

```
$HOME/sqllib/samples/pd/db2_hang_analyze -db sample -log
Invoked: /home/hotel32/shenli/sqllib/samples/pd/db2_hang_analyze -db sample -log
APPLICATION HANG DETECTION: Started on Fri Jan 25 14:41:38 EST 2013
Sleeptime                :  60 seconds
Timer Limit              :  300 seconds
Node Member              :  default
Retry Limit              :  3
Log                      :  yes
SQL                      :  no
Event Metrics Available  :  yes
Logfile                  :  db2_hang_analyze.20130125.14.41.38.10297.log
Script PID               :  10297
CPU Threshold            :  0.1%
Post Detection Script    :  none
Path                     :  /home/hotel32/shenli/sqllib/db2dump

Press CTRL-C or CTRL-Z to terminate script
Pre-loop setup...
Iteration 1: No hang found.
Iteration 2: No hang found.
Iteration 3: No hang found.
Iteration 4: No hang found.
Iteration 5: POSSIBLE HANG DETECTED!
Logfile                  :  /home/hotel32/shenli/sqllib/db2dump/db2_hang_analyze.20130125.14.41.38
VIEW REPORT AT           :  /home/hotel32/shenli/sqllib/db2dump/db2_hang_analyze.20130125.14.41.38
APPLICATION HANG DETECTION: Ended on Fri Jan 25 14:46:47 EST 2013
```

If a hang or multiple hangs are detected, then a report file is generated that lists the hanging applications:

```
cat /home/hotel32/shenli/sqllib/db2dump/db2_hang_analyze.20130125.14.41.38.10297.report
APPLICATION HANG DETECTION: Started on Fri Jan 25 14:41:38 EST 2013
Sleeptime                :  60 seconds
Timer Limit              :  300 seconds
Node Member              :  default
Retry Limit              :  3
Log                      :  yes
SQL                      :  no
Event Metrics Available  :  yes
Logfile                  :  db2_hang_analyze.20130125.14.41.38.10297.log
Script PID               :  10297
CPU Threshold            :  0.1%
Post Detection Script    :  none
Path                     :  /home/hotel32/shenli/sqllib/db2dump/

POTENTIAL APPLICATIONS HANGING: 3 application(s).
Apphdl             :  7
Status             :  CommitActive
AgentEDUID         :  16

Apphdl             :  9
Status             :  CommitActive
AgentEDUID         :  28

Apphdl             :  19
Status             :  CommitActive
AgentEDUID         :  37


APPLICATION HANG DETECTION: Ended on Fri Jan 25 14:46:47 EST 2013
```

Example 2

You can control how the script detects possible hanging application by altering a number of the options:

```
$HOME/sqllib/samples/pd/db2_hang_analyze -db sample -member 0 -log -timerlimit 30 -sleeptime 15 -sql
Invoked: /home/hotel32/shenli/sqllib/samples/pd/db2_hang_analyze -db sample -member 0 -log -timerlim
APPLICATION HANG DETECTION: Started on Fri Jan 25 15:38:27 EST 2013
Sleeptime               :  15 seconds
Timer Limit             :  30 seconds
Node Member             :  0
Retry Limit             :  3
Log                     :  yes
SQL                     :  yes
Event Metrics Available :  yes
Logfile                 :  db2_hang_analyze.20130125.15.38.27.10189.log
Script PID              :  10189
CPU Threshold           :  0.1%
Post Detection Script   :  none
Path                    :  /TMP

Press CTRL-C or CTRL-Z to terminate script
Pre-loop setup...
Iteration 1: No hang found.
Iteration 2: POSSIBLE HANG DETECTED!
Logfile                 :  /TMP/db2_hang_analyze.20130125.15.38.27.10189.log
VIEW REPORT AT          :  /TMP/db2_hang_analyze.20130125.15.38.27.10189.report
APPLICATION HANG DETECTION: Ended on Fri Jan 25 15:39:03 EST 2013
```

In this example, the user wants to check whether there are any applications that are hanging on member 0. A number of **db2pd** command metrics are gathered every 15 seconds, instead of the default 60 seconds. The application is identified as hanging if its metrics do not change within 30 seconds, instead of the default of 300 seconds. After a hang is detected, the latest SQL statement that relates to the hanging application is printed in the report file. The report and log files are written to the /TMP/ directory.

The following report file displays the results. One application is possibly hanging.

```
cat /TMP/db2_hang_analyze.20130125.15.38.27.10189.report
APPLICATION HANG DETECTION: Started on Fri Jan 25 15:38:27 EST 2013
Sleeptime               :  15 seconds
Timer Limit             :  30 seconds
Node Member             :  0
Retry Limit             :  3
Log                     :  yes
SQL                     :  yes
Event Metrics Available :  yes
Logfile                 :  db2_hang_analyze.20130125.15.38.27.10189.log
Script PID              :  10189
CPU Threshold           :  0.1%
Post Detection Script   :  none
Path                    :  /TMP

POTENTIAL APPLICATIONS HANGING: 1 application(s).
Apphdl              :  51
Status              :  UOW-Executing
AgentEDUID          :  44
Current query       :  select * from staff
Last query:         :  none


APPLICATION HANG DETECTION: Ended on Fri Jan 25 15:39:03 EST 2013
```

Usage notes

The script does not consider applications that are in a lock wait state to be hanging.

The script requires Perl v5.6.0 or higher.

db2_hang_detect - Detect, report on, and resolve Db2 database hangs script:

Detects possible application hangs on a Db2 database and acts to inform and resolve the situation. **`db2_hang_detect`** is a ksh script that contains a package of hang detection tests. The **`db2_hang_detect`** script is available only on Linux and UNIX operating systems.

The **`db2_hang_detect`** scriptd is designed to run periodically per node and per instance. Ideally, you run the script as a complement to the high availability (HA) capabilities that are provided by an HA cluster manager.

The **`db2_hang_detect`** script is in the `sqllib/samples/pd/` directory.

There are four hang detection tests which, depending on the value of the parameters that you choose, run separately or as groups:

instance_level_detect
Detects an instance that is hanging.

latch_level_detect
Detects hangs that are caused by latches.

db_cat_node_fail_monitor
Detects hangs during node recovery.

progress_detect
Detects hangs in agents.

Authorization

Root user authority

Required connection

None

Syntax

```
db2_hang_detect [DB2INSTANCE] [NN] [DETECTLEVEL] [ACTION] [DBNAME] [NOTIFYADDRESS] [LOOPBACKNAME]
```

Parameters

`DB2INSTANCE`
Specifies the instance to monitor.

`NN` *partition or node number*
Specifies the number of the node or partition to monitor.

`DETECTLEVEL` *test level*
Specifies the level of the hang detection test. The following values are possible:

`0` Runs none of the tests.

`1` Runs instance_level_detect test.

`2` Runs instance_level_detect and latch_level_detect tests.

`3` Runs instance_level_detect, latch_level_detect, and db_cat_node_fail_monitor tests.

4 Runs instance_level_detect, latch_level_detect, db_cat_node_fail_monitor, and progress_detect tests.

ACTION
Specifies the action that is taken if a hang is detected. The following values are possible:

INFORM
Sends a message that specifies which application is hanging.

FORCE
Stops the application that is hanging by forcing it off the system.

TERMINATE
Stops the application that is hanging by terminating it.

DBNAME
Specifies the name of the database to monitor.

NOTIFYADDRESS
Specifies the email address to which a message is sent if a hang is detected.

LOOPBACKDBNAME
Specifies the name of the database to catalog through a remote connection. If you specify `nulldb`, no database is cataloged.

VERBOSE
Prints status messages to standard output. The following values are possible:

VERBOSE
Prints the status messages.

NOVERBOSE
Does not print status messages.

Example

In the following example, you search for a potential hang on instance DB2INST1. You specify node 0 and user all four tests. If a hang is detected, you are informed by a message. The database that is being monitored is DB2SAMPL. The message that an application is hanging is sent to the email address user@domain.com. No database is cataloged and no status message is printed.

```
db2_hang_detect db2inst1 0 4 inform  db2sampl user@domain.com nulldb noverbose
```

Return codes

The following table lists all possible return codes.

Table 9. db2_hang_detect command return codes

Return code	Explanation
0	Indicates that no hang is detected and that the instance is up.
1	Indicates that the instance is offline.
2	Indicates that an environmental issue, for example your system is not set up properly, is preventing a successful execution.
3 and higher	Indicates a potential hang.

Usage notes

You must run the script as root. You must also run the script locally on the server where the Db2 instance is located.

Multiple copies of the script cannot be run concurrently.

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 IBM Software Support. The process of taking a trace entails setting up the trace facility, reproducing the error and collecting the data.

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

Collecting a trace often has a detrimental effect on the performance of a Db2 instance. The degree of performance degradation is dependent on the type of problem and on how many resources are being used to gather the trace information.

IBM Software 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 IBM Software 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 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 recreated, 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 information:

Video: Db2 and CLI Tracing

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 additional processor usage when a trace is running so enabling the trace facility might impact your system's performance.

Note: On Windows operating systems, the **db2trc** command will trace all Db2 instances that belong to the same installed copy. On Linux and UNIX operating systems, Db2 instances can be traced individually.

In general, IBM Software Support and development teams use Db2 traces for troubleshooting. You might run a trace to gain information about a problem that you are investigating, but its use is rather limited without knowledge of the Db2 source code.

Nonetheless, it is important to know how to correctly turn on tracing and how to dump trace files, just in case you are asked to obtain them.

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

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

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

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

```
db2trc on -u
```

This will provide information about all of the additional options (labeled as "facilities") that can be specified when turning on a Db2 trace.

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

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

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

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

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

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

1. On multiple database partition systems, you must run a trace for each physical (as opposed to logical) database partition.
2. On HP-UX, Linux and Solaris platforms, if the trace is turned off after the instance has been started, a very small buffer will be used the next time the trace is started regardless of the size specified. For example, yesterday you turned trace on by using **db2trc on -l 8m**, then collected a trace, and then turned the trace off (**db2trc off**). Today you want to run a trace with the memory buffer set for 32 megabytes (**db2trc on -l 32m**) without bringing the instance down and restarting. You will find that in this case trace will only get a small buffer. To effectively run a trace on these platforms, turn the trace on before starting the instance with the size buffer you need and "clear" the buffer as necessary afterwards.

To reduce the amount of data collected or formatted, the **db2trc** command supports several mask options. Reducing the amount of data collected is useful, because it can reduce the additional processor usage incurred due to an ongoing trace collection, and because you can collect data more selectively. Collecting data more selectively can also help speed up problem diagnosis.

You typically use the **-m** mask option under the guidance of IBM support. However, you can use the **-p** mask option to collect a trace only for specific process IDs (and optionally thread IDs). For example, to enable tracing for process 77 with threads 1, 2, 3, and 4, and for process 88 with threads 5, 6, 7, and 8 the syntax is:

```
db2trc on -p 77.1.2.3.4,88.5.6.7.8
```

Setting and Capturing db2trc on Mac

On most of the Mac systems, by default, the Kernel maximum shared memory that is set is 4 MB. The **db2trc** can't be turned on for such systems. The shared memory must be set to at-least 64 MB for turning on the **db2trc**.

Use the following command to see the current Kernel shared memory settings:

```
sysctl -A | grep shm
```

To modify the Kernel shared memory, create a **sysctl.conf** in /etc directory that uses super user privilege.

```
sudo vi /etc/sysctl.conf
```

A sample **sysctl.conf** file with maximum shared memory setting to support 64 MB would look like:

```
kern.sysv.shmmax: 67108864
kern.sysv.shmmin: 1
kern.sysv.shmmni: 512
kern.sysv.shmseg: 128
kern.sysv.shmall: 16384
machdep.pmap.hashmax: 14
security.mac.posixshm_enforce: 1
security.mac.sysvshm_enforce: 1
```

Once the changes are done, restart the system for the changes to take effect.

Using the trcon and troff scripts to control trace collection

Two scripts are available to make trace collection simpler by replacing several manually issued commands with a single script invocation.

The **db2trcon** script turns on tracing and supports several options. You can use this script to turn on db2trc for a specified amount of time, specify to collect a trace for only the top processor time consuming engine dispatchable units (EDUs), and generate dump, flow and format files automatically. For example, to turn tracing on for a duration of 45 seconds for the top 5 processor time consuming EDUs, sampled in 15 second intervals from the time the script is run, issue the following command:

```
db2trcon
-duration 45 -top 5 -interval 15 -flw -fmt
```

When db2trc is turned off after the specified duration, db2trcon automatically generates the dump, flow and format files for you.

The **db2trcoff** turns off tracing and can generate dump, flow and format files automatically with a single command. For example, to turn db2trc off with -force and generate flow format and dump files, issue the following command:

```
db2trcoff -flw
-fmt -force
```

Note that if you turned tracing on with the db2trcon script and specified a duration, you do not need to issue the db2troff command separately.

Dumping a Db2 trace file:

After the trace facility has been enabled using the `ON` option, all subsequent work done by the instance will be traced.

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

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

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

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

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

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

Formatting a Db2 trace:

The dump file created by the **db2trc dmp** command is in binary format and is not readable. You can format the binary dump file into a readable file to show the flow control and send the formatted output to a null device.

One of the ways to format the binary file is by issuing the **db2trc** command with the `flow` option, as shown in the following example:

```
db2trc flow example.trc nul
```

where `example.trc` is the binary file.

If there was a problem reading the file and the trace output was wrapped, these would be reflected in the output for this command.

At this point, you can send the dump file to IBM Support. IBM Support will format the file based on your Db2 service level. However, you might be asked to format the dump file into an ASCII file before sending it. You accomplish this by using the `flow` and `format` options. You must provide the name of the binary dump file and the name of the ASCII file that you want to create, as shown in the following example:

```
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 the output shows that the value of `Trace wrapped` is `YES`, the trace buffer was not large enough to contain all the information that was collected during the trace period. A wrapped trace might be acceptable, depending on the situation. If you are interested in the most recent information (the information that is maintained unless you specified the `-i` option), what is in the trace file might be sufficient. However, if you are interested in what happened at the beginning of the trace period or if you are interested in everything that occurred, you might want to redo the operation with a larger trace buffer.

There are options for formatting a binary file. For example, you can issue `db2trc format -xml trace.dmp trace.fmt` to convert the binary data into an XML format that can be parsed. You can also use the **formattedFlow** parameter of the **db2trc** command to parse the binary file into a formatted text file that is organized in chronological order. You can also create a performance report from a dump file by using the **perfrep** parameter. Additional options are shown in the description of the **db2trc** command.

On Linux and UNIX operating systems, if a severe error occurs, the Db2 software automatically dumps the trace buffer to disk when it shuts down the instance due to a severe error. If tracing is enabled when an instance ends abnormally, a file is created in the diagnostic directory. The file name is `db2trdmp.`*nnn*, where *nnn* is the database partition number. The file creation in the diagnostic directory does not occur on Windows operating systems when an instance ends abnormally due to an error. You must dump the trace manually.

To summarize, the following example shows 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
```

Parsing a Db2 trace into readable text:

You can format binary dump files from the **db2trc** command into a text file by using the **formattedFlow** or **fflw** parameter.

This parameter is used to format a binary trace dump file into text (similar to **flow** command), sorted in chronological order instead of being grouped by process ID (PID). The **formattedFlow** command allows merging of the flow format records into one group. It also enables each record to carry the original process ID (PID) or thread ID (TID), engine dispatchable unit (EDU) name, and member (node) number. The **formattedFlow** then sorts the records by the record ID in chronological order. This allows a global and chronological view of what happened in the entire Db2 system.

Example

The following command creates formatted output in the `out.fflw` file:

```
db2trc formattedFlow trc.dmp out.fflw
```

The sample output is as follows:

```
PID-TID      EduName Node RecordNum Function
[...]
12648456-258 db2wdog [ 0] 19735     |||cryptContextInit entry
12648456-258 db2wdog [ 0] 19736     |||cryptContextInit data [probe 10]
12648456-258 db2wdog [ 0] 19737     |||cryptContextInit data [probe 100]
12648456-258 db2wdog [ 0] 19738     |||cryptContextInit exit
12648456-258 db2wdog [ 0] 19739     ||sqloWatchDogSetup data [probe 2]
12648456-258 db2wdog [ 0] 19740     ||sqloWatchDogSetup exit
12648456-258 db2wdog [ 0] 19741     ||sqlogmblkEx entry
12648456-258 db2wdog [ 0] 19742     |||sqloGetPrivatePoolHandle entry
12648456-258 db2wdog [ 0] 19743     |||sqloGetPrivatePoolHandle exit
11731144-258 db2sysc [ 0] 19744     sqloGetEnvInternal entry
11731144-258 db2sysc [ 0] 19745     sqloGetEnvInternal exit [rc = 0x870F0104 = -2029059836 = RC_ENV_NOT_FOUND]
12648456-258 db2wdog [ 0] 19746     ||sqlogmblkEx mbt [Marker:PD_OSS_ALLOCATED_MEMORY ]
12648456-258 db2wdog [ 0] 19747     ||sqlogmblkEx exit
11731144-258 db2sysc [ 0] 19748     sqloSystemControllerMain entry
11731144-258 db2sysc [ 0] 19749     |sqloChangeProcessName entry
11731144-258 db2sysc [ 0] 19750     |sqloChangeProcessName data [probe 5]
11731144-258 db2sysc [ 0] 19751     |sqloChangeProcessName exit
11731144-258 db2sysc [ 0] 19752     |sqloGetShrEDUWaitElem entry
11731144-258 db2sysc [ 0] 19753     ||sqlo_waitlist::initialize entry
11731144-258 db2sysc [ 0] 19754     ||sqlo_waitlist::initialize exit
11731144-258 db2sysc [ 0] 19755     ||sqlogmblkEx entry
11731144-258 db2sysc [ 0] 19756     ||sqlogmblkEx mbt [Marker:PD_OSS_ALLOCATED_MEMORY ]
11731144-258 db2sysc [ 0] 19757     ||sqlogmblkEx exit
11731144-258 db2sysc [ 0] 19758     |sqloGetShrEDUWaitElem data [probe 10]
11731144-258 db2sysc [ 0] 19759     |sqloGetShrEDUWaitElem data [probe 20]
11731144-258 db2sysc [ 0] 19760     |sqloGetShrEDUWaitElem exit
12648456-258 db2wdog [ 0] 19761     |sqloRunInstance data [probe 2]
12648456-258 db2wdog [ 0] 19762     |sqloRunInstance exit
11731144-258 db2sysc [ 0] 19763     |sqloGetKernelThreadIDFromEDUID entry [eduid 258 eduname db2sysc]
```

Performance reports for Db2 traces:

You can create a performance report for a Db2 trace by using the **db2trc** command with the **perfrep** parameter.

You can issue the **db2trc** command with the **perfrep** parameter for dump files that you created by issuing the **db2trc** command with the `-t` option. The new feature eliminates the excessive processor usage that was previously caused by executing a performance trace in addition to the regular trace for diagnostic purposes.

The **perfrep** report displays the time that was spent in each caller function, in addition to the elapsed time that was spent to execute the trace. You can sort the output by the number of invocations, the time that was spent in each function, or the elapsed time that was spent, in ascending or descending order. By default, the output is sorted by the time that was spent in each function, in descending order. You can also group the output by using the combination of member (node) number, process ID (PID), and thread ID (TID).

Example

In the following example, the `db2trc` command is run with the **perfrep** parameter:

```
db2trc perfrep trc.dmp perfrep.out
```

Sample output in the `perfrep.out` file is as follows:

```
nCalls TotalElapsed   AvgElapsed   TotalSpent     AvgSpent     FunctionName
30     128.900145590  4.296671520  128.900122035  4.296670735  sqlorqueInternal
22     35.070790711   1.594126850  35.070442569   1.594111026  sqlo_waitlist::timeoutWait
5      33.333113807   6.666622761  33.333113807   6.666622761  OSSHIPCWaitpost::wait
6      20.412543399   3.402090567  20.412527985   3.402087997  sqlorest
1      18.300280961   18.300280961 18.300280961   18.300280961 sqloAlarmThreadEntry
3      18.248856592   6.082952197  18.248856592   6.082952197  sqloReadNamedPipe
1      14.907893602   14.907893602 14.907893602   14.907893602 OSSHIPCSemaphore::wait
1      14.888266323   14.888266323 14.888030081   14.888030081 sqloSSemP
7      8.131142984    1.161591855  8.130613854    1.161516265  sqlowchd
4      5.736910141    1.434227535  5.736471457    1.434117864  sqloWaitIPCWaitPost
1      21.967713234   21.967713234 3.600555023    3.600555023  sqleSysCtlr
115    1.111718480    0.009667117  1.105887133    0.009616410  sqlnlsgetcpcc
34     0.701106437    0.020620778  0.701106437    0.020620778  OSSHLibrary::load
1      0.157144687    0.157144687  0.143064433    0.143064433  cryptContextRealInit
4      0.098807890    0.024701972  0.093563598    0.023390899  sqloLoadModule
25     0.131057287    0.005242291  0.079349292    0.003173972  NetlsRequestLicense
3245   0.093867246    0.000028927  0.067123446    0.000020685  hregReadBlock
97     0.062272831    0.000641988  0.061129273    0.000630199  sqlnlscmsg
3      0.068937667    0.022979222  0.058964884    0.019654961  sqloexec
3      0.057550155    0.019183385  0.052918398    0.017639466  sqkfDynamicResourceMgr::InitResource
```

DRDA trace files:

Before analyzing DRDA traces, you must 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 Power Systems™ Servers database server).

Trace utility:

The **db2drdat** utility records the data that is interchanged between the Db2 Connect server (on behalf of the IBM data server client) and the IBM mainframe database server.

As a database administrator (or application developer), you might find it useful to understand how this flow of data works, because this knowledge can help you determine the origin of a particular problem. Suppose you found yourself in the following situation: you issue a `CONNECT TO` database statement for a IBM mainframe database server but the command fails and you receive an unsuccessful return code. If you understand exactly what information was conveyed to the IBM mainframe 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 IBM mainframe database server management system. Data sent to the IBM mainframe database server is labeled SEND BUFFER and data received from the IBM mainframe 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 IBM mainframe database server. The send and receive buffers are arranged from the oldest to the most recent within the file. Each buffer has:

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

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

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

Trace output:

The trace file that is written by the db2drdat utility contains operational information about DRDA.

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

- -r
  - Type of DRDA reply/object
  - Receive buffer
- -s

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

Note:

1. A value of zero for the exit code indicates that the command completed successfully, and a non-zero value indicates that it did not.
2. The fields returned vary based on the API used.
3. The fields returned vary based on the platform on which Db2 Connect is running, even for the same API.
4. If the **db2drdat** command sends the output to a file that already exists, the old file will be erased unless the permissions on the file do not allow it to be erased.

Trace output file analysis:

The db2trc output file displays information about operations within the database and is used to troubleshoot the database.

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 IBM mainframe database server CCSID(s)
- The IBM mainframe 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 IBM mainframe 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 IBM mainframe 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 about the IBM mainframe database management system. It consists of a 20-byte application ID followed by an 8-byte process ID (or 4-byte process ID and 4-byte thread ID). It is represented by code point X'115E', and in this example its value is `db2bp` padded with blanks followed by `000C50CC`. On a Linux or UNIX IBM data server client, this value can be correlated with the **ps** command, which returns process status information about active processes to standard output.

ACCRDB

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

The accounting string has code point X'2104'.

The code set configured for the Db2 Connect workstation is shown by locating the CCSID object CCSIDSBC (CCSID for single-byte characters) with code point X'119C' in the **ACCRDB** command. In this example, the CCSIDSBC is X'0333', which is 819.

The additional objects CCSIDDBC (CCSID for double-byte characters) with code points X'119D' and CCSIDMBC (CCSID for mixed-byte characters) with code point X'119E', 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.

EXCSATRD and ACCRDBRM

CCSID values are also returned from the IBM mainframe 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 IBM mainframe 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 IBM mainframe database server, SQLCODE -332 will be returned to the user with the source and target code pages. If the IBM mainframe 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 for z/OS®. The version number is also indicated. ARI is Db2 Server for VSE & VM, SQL is Db2 database or Db2 Connect, and QSQ is IBM Db2 for IBM i.

Trace output file samples:

When running a DRDA trace, you receive an output file with information about current DRDA operations in the database. You can use this information to troubleshoot your database.

The following figures show sample output illustrating some DRDA data streams exchanged between Db2 Connect workstations and a host or System i® database server. From the user's viewpoint, a `CONNECT TO` database command has been issued using the command line processor (CLP).

Figure 1 on page 102 uses Db2 Connect Enterprise Edition Version 9.1 and Db2 for z/OS Version 8 over a TCP/IP connection.

```
1 data Db2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 0 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 233
2 data Db2 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):

            EXCSATT RQSDSS                   (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  00C3D041000100BD 1041007F115E8482   ...A.....A...^..  .C}........".;db
  0010  F282974040404040 4040404040404040   ...@@@@@@@@@@@@@  2bp
  0020  4040F0F0F0C3F5F0 C3C3F0F0F0000000   @@..............    000C50CC000...
  0030  0000000000000000 0000000000000000   ................  ................
  0040  0000000000000000 000000000060F0F0   .............`..  .............-00
  0050  F0F1A2A495404040 4040404040404040   .....@@@@@@@@@@@  01sun
  0060  4040404040404040 4040404040404040   @@@@@@@@@@@@@@@@
  0070  C4C5C3E5F8404040 F0A2A49540404040   .....@@@....@@@@  DECV8   0sun
  0080  4040404040404040 4000181404140300   @@@@@@@@@.......           .......
  0090  0724070008147400 05240F0008144000   .$....t..$....@.  .............. .
  00A0  08000E1147D8C4C2 F261C1C9E7F6F400   ....G....a......  .....QDB2/AIX64.
  00B0  08116D9481A29200 0C115AE2D8D3F0F9   ..m.......Z.....  .._mask...]SQL09
  00C0  F0F0F0                              ...               000

            ACCSEC RQSDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0026D00100020020 106D000611A20003   .&..... .m......  ..}......_...s..
  0010  00162110E2E3D3C5 C3F1404040404040   ..!.......@@@@@@  ....STLEC1
  0020  404040404040                        @@@@@@
3 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110546200 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 105
4 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110549755 probe 1178
 bytes 122

 RECEIVE BUFFER(AR):

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

            ACCSECRD OBJDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0010D0030002000A 14AC000611A20003   ................  ..}..........s..
5 data Db2 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)

```
6 data Db2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 110659711 probe 1177
 bytes 250

 SEND BUFFER(AR):

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

            ACCRDB RQSDSS                       (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  00ADD001000200A7 20010006210F2407   ........ ...!.$.  ..}....x........
  0010  00172135C7F9F1C1 F0C4F3C14BD7C1F8   ..!5........K...  ....G91A0D3A.PA8
  0020  F806030221064600 162110E2E3D3C5C3   ....!.F..!......  8..........STLEC
  0030  F140404040404040 4040404040000C11   .@@@@@@@@@@@@...  1             ...
  0040  2EE2D8D3F0F9F0F0 F0000D002FD8E3C4   ............/...  .SQL09000....QTD
  0050  E2D8D3C1E2C30016 00350006119C0333   .........5.....3  SQLASC..........
  0060  0006119D04B00006 119E0333003C2104   ...........3.
7 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259908001 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 176
8 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 0 nsec 259911584 probe 1178
 bytes 193

 RECEIVE BUFFER(AR):

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

            ACCRDBRM RPYDSS                     (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  009BD00200020095 2201000611490000   ........"....I..  ..}....n........
  0010  000D002FD8E3C4E2 D8D3F3F7F0000C11   .../............  ....QTDSQL370...
  0020  2EC4E2D5F0F8F0F1 F500160035000611   ............5...  .DSN08015.......
  0030  9C04B80006119E04 B80006119D04B000   ................  ................
  0040  0C11A0D5C5E6E3D6 D540400006212524   .........@@..!%$  ...NEWTON  .....
  0050  34001E244E000624 4C00010014244D00   4..$N..$L....$M.  ....+...<.....(.
  0060  06244FFFFF000A11 E8091E768301BE00   .$O.........v....  ..!.....Y...c...
  0070  2221030000000005 68B3B8C7F9F1C1F0   "!......h.......  ...........G91A0
  0080  C4F3C1D7C1F8F840 4040400603022106   .......@@@@...!.  D3APA88    .....
  0090  46000A11E8091E76 831389             F......v...       ....Y...c.i
9 data Db2 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 2. Example of Trace Output (TCP/IP connection) continued

```
10 data Db2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 364440751 probe 1177
 bytes 27

 SEND BUFFER(AR):

            RDBCMM RQSDSS                     (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200E               ........ .        ..}.......
11 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475009631 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54
12 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 2 nsec 475014579 probe 1178
 bytes 71

 RECEIVE BUFFER(AR):

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

            SQLCARD OBJDSS                    (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000BD00300010005 2408FF             ........$..       ..}........
13 data Db2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721710319 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 126
14 data Db2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 721727276 probe 1177
 bytes 143

 SEND BUFFER(AR):

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

            SQLSTT OBJDSS                     (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  002BD00300010025 2414000000001B64   .+.....%$......d  ..}.............
  0010  656C657465206672 6F6D206464637375   elete from ddcsu  .%......?_......
  0020  73312E6D79746162 6C65FF             s1.mytable.       ..._`./.%..
15 data Db2 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 3. Example of Trace Output (TCP/IP connection) continued

```
16 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.1178)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 832906528 probe 1178
 bytes 119

 RECEIVE BUFFER(AR):

            SQLCARD OBJDSS                  (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  0066D00300010060 240800FFFFFF3434   .f.....`$.....44  ..}....-........
  0010  3237303444534E58 4F544C2000FFFFFE   2704DSNXOTL ....  ......+.!.<.....
  0020  0C00000000000000 00FFFFFFFF000000   ................  ................
  0030  0000000000572020 2057202020202020   .....W   W        ................
  0040  001053544C454331 2020202020202020   ..STLEC1          ....<...........
  0050  2020000F44444353 5553312E4D595441     ..DDCSUS1.MYTA  ............(...
  0060  424C450000FF                        BLE...            .<....
17 data Db2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833156953 probe 100
 bytes 16

 Data1  (PD_TYPE_UINT,8) unsigned integer:
 10
18 data Db2 DRDA Communication Manager sqljcSend fnc (3.3.54.5.0.1177)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 833159843 probe 1177
 bytes 27

 SEND BUFFER(AR):

            RDBRLLBCK RQSDSS                (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
  0000  000AD00100010004 200F               ........ .        ..}.......
19 data Db2 DRDA Communication Manager sqljcReceive fnc (3.3.54.3.0.100)
 pid 807116 tid 1 cpid -1 node 0 sec 5 nsec 943302832 probe 100
 bytes 12

 Data1  (PD_TYPE_UINT,4) unsigned integer:
 54
20 data Db2 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 4. Example of Trace Output (TCP/IP connection) continued

Subsequent buffer information for DRDA traces:

You can analyze subsequent send and receive buffers for additional information.

If the next request contains a commit, the **`commit`** command instructs the IBM mainframe database server management system to commit the current unit of work. The fourth buffer is received from the IBM mainframe 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 102 shows an example of a receive buffer containing an error SQLCA at trace entry 16.

Obtaining traces of applications that use the IBM Data Server Driver for JDBC and SQLJ:

You can enable IBM Data Server Driver for JDBC and SQLJ tracing using Connection or DataSource properties, driver global configuration properties, or DB2TraceManager methods.

About this task

For details, see .

CLI traces:

The CLI trace contains a record of all the CLI function calls that the CLI driver made.

The CLI trace is an essential tool for diagnosing problems with applications that access the CLI driver. The CLI trace provides diagnostic information when a problem is encountered in any of the following places:
- A CLI application
- An ODBC application, because ODBC applications use the CLI interface to access IBM database servers
- A CLI stored procedure

By default, the trace utility is disabled. When enabled, the trace utility generates one or more trace files whenever an application accesses the CLI driver. These trace files provide the following information:
- The order in which the application called the CLI functions
- The contents of input and output parameters that were passed to and received from the CLI functions
- The return codes and any error or warning messages that the CLI functions generated

CLI trace file analysis provides a number of benefits. First, subtle program logic and parameter initialization errors are often evident in the traces. Second, CLI traces might suggest ways of better tuning an application or the databases that it accesses. For example, if a CLI trace shows that a particular set of columns is queried many times, you might want to create an index corresponding to one of the columns to improve application performance. Finally, analysis of CLI trace files can help you understand how a third-party application or interface is behaving.

Different methods for tracing the CLI driver

You can trace the CLI driver by using one of the following methods:

- By issuing the **db2trc on -cli** command. The **db2trc on -cli** command provides a dynamic way to obtain a CLI trace. You can enable and disable the trace without restarting the application. The command creates a binary trace file that you must process into user-readable output by issuing the **db2trc fmt -cli** *trace.dmp clitrace.txt* command, where *trace.dmp* is the name of the binary trace file and *clitrace.txt* is the name of the output file.
- By modifying the `db2cli.ini` configuration file to include the CLI trace configuration keywords that controls the trace. If you use the `db2cli.ini` file to obtain a trace, you must restart the application for the CLI trace configuration keywords in the `db2cli.ini` configuration file to take effect. The restart is required because the values of the CLI trace configuration keywords are read from the `db2cli.ini` file only when an application is initialized. You can use a special CLI trace keyword, **TraceRefreshInterval**, to indicate an interval at which the values of specific CLI trace keywords are reread from the `db2cli.ini` file.

  You can override the **TRACE** keyword setting in the `db2cli.ini` file by setting the **SQL_ATTR_TRACE** environment attribute with the **SQLSetEnvAttr()** API in the application.

CLI driver traces versus ODBC driver manager traces

When diagnosing ODBC applications, it is often easiest to determine the problem by using an ODBC trace or a CLI trace. An ODBC driver manager provides the capability to enable an ODBC trace. An ODBC driver manager is not part of a Db2 product; you must acquire an ODBC driver manager from an independent software vendor. To determine how to enable ODBC tracing, consult your ODBC driver manager documentation.

It is important to understand the difference between an ODBC driver manager trace and a CLI driver trace. An ODBC driver manager trace shows the ODBC function calls that an ODBC application makes to the ODBC driver manager. By contrast, a CLI driver trace shows the function calls that the ODBC driver manager makes to the CLI driver on behalf of the application.

The CLI traces are specific to the Db2 software and typically contain more information than a generic ODBC trace. Both types of traces list entry and exit points for all ODBC function calls from an application and include any parameters that are passed to the ODBC functions and return codes from the ODBC functions.

An ODBC driver manager might forward some function calls directly from the application to the CLI driver. However, the ODBC driver manager might also delay or avoid forwarding some function calls to the driver. The ODBC driver manager might also modify application function arguments or map application functions to other functions before forwarding the call to the CLI driver.

The ODBC driver manager intervenes to perform listed tasks:
- ODBC 2.0 functions that are deprecated in ODBC 3.0 are mapped to new functions.
- ODBC 2.0 function arguments that are deprecated in ODBC 3.0 are mapped to equivalent ODBC 3.0 arguments.
- The Microsoft cursor library maps calls to functions such as the `SQLExtendedFetch()` function to multiple calls to the`SQLFetch()` function and other supporting functions to achieve the same result.

- ODBC driver manager connection pooling usually defers `SQLDisconnect()` function call requests or avoids them altogether if the connection is reused.

For you to determine what ODBC function calls are made to the CLI driver by the ODBC driver manager, you might find an ODBC driver manager trace to be a useful complement to the CLI driver trace.

For more information about the ODBC driver manager trace, see the ODBC driver manager documentation that is provided by the independent software vendor. For more information about the Microsoft ODBC 3.0 Software Development Kit and Programmer's Reference, see http://www.msdn.microsoft.com/.

CLI driver and Db2 traces

The CLI driver uses many internal Db2 functions to do its work. These internal Db2 function calls are logged in the Db2 traces. The Db2 traces are meant to assist IBM Service in problem determination and resolution.

CLI traces and CLI stored procedures

On all workstation platforms, you can use the CLI trace utility to trace the execution of CLI stored procedures.

The CLI trace information and instructions are generic and apply to the stored procedures. However, unlike applications that are typically executed on a remote computer that is separate from the database server, stored procedures execute on the database server. Therefore, you must take following steps when tracing the CLI stored procedures:
- Specify the trace keyword options in the `db2cli.ini` file on the Db2 server.
- You must set all keywords correctly before you run the **`db2start`** command to start the database manager.

Note: The **`TRACEREFRESHINTERVAL`** and **`QUERYTIMEINTERVAL`** CLI keywords are ignored if you use them inside a stored procedure that uses CLI API calls.

Related information:

Video: Db2 and CLI Tracing

Tracing the CLI driver by using the **`db2trc on -cli`** *command:*

You can trace the CLI function calls dynamically with the **`db2trc on -cli`** command.

About this task

The formatted Db2 trace file that is generated with the **`db2trc fmt -cli`** command is similar to that of the CLI trace. By using the **`db2trc on -cli`** command, you can:
- Dynamically enable and disable the trace without having to restart the application.
- Specify the trace output path and file name.
- Trace a specific process ID or list of process IDs.
- You can obtain the details of all statement handles that were allocated by the CLI driver prior to enabling the Db2 trace with the `-dumpstmt` option.

- You can format the trace into separate text files that are based on the thread ID of a process if an existing directory is specified for the trace format command.

Important: Do not run the trace unless you must diagnose a problem. Tracing reduces the overall performance of your system.

Procedure

1. Enable the CLI trace by issuing one of the following commands:
   - To trace the CLI driver activities into the trace memory buffer:
     ```
     db2trc on -cli
     ```
   - To trace the CLI driver activities into a binary trace dump file:
     ```
     db2trc on -cli -f dumpFile
     ```
     where *dumpFile* is the name of the binary dump file. You can qualify the file name with the directory path, as shown in the following example:
     ```
     db2trc on -cli -f c:\temp\clitrace\dump.file
     ```
     If you do not qualify the file name with the directory path, the file is generated in the path where you issue the **db2trc** command.
   - To trace only one specific process ID:
     ```
     db2trc on -cli -p pid
     ```
     To trace only one specific process ID to a file:
     ```
     db2trc on -cli -p pid -f dumpFile
     ```
     To trace multiple specific process IDs:
     ```
     db2trc on -cli -p pid,pid,...
     ```
     To trace multiple specific process IDs to a file:
     ```
     db2trc on -cli -p pid,pid,... -f dumpFile
     ```
     where *pid* is the process ID of a CLI application, as shown in the following example:
     ```
     db2trc on -cli -p 6860,6888 -f c:\temp\clitrace\dump.file
     ```
   - To trace the CLI driver activities and obtain the details of all statement handles that were allocated by the CLI driver prior to enabling Db2 trace, issue one of the following commands:
     ```
     db2trc on -cli -dumpstmt
     db2trc on -cli -dumpstmt -f dumpFile
     ```
2. Run the application scenario that you want to investigate.
3. Turn off the tracing in one of the following ways.
   - If you enabled the trace without specifying the **-f** *dumpFile* parameter, issue the following commands:
     ```
     db2trc dump dumpFile
     db2trc off
     ```
     where *dumpFile* is the name of the binary trace dump file. You can qualify the file name with the directory path.
   - If you enabled the trace by specifying the **-f** *dumpFile* parameter, issue the following command:
     ```
     db2trc off
     ```
4. Format the binary file into a CLI trace by issuing the following command:
   ```
   db2trc fmt -cli CLItraceName
   ```

CLItraceName can be one of the following objects:

- The name of the formatted trace file that is to be created.

  ```
  db2trc fmt –cli c:\temp\clitrace\dump.file clitrcfile.txt
  ```
- The name of the formatted trace file with the path name.

  ```
  db2trc fmt –cli c:\temp\clitrace\dump.file c:\temp\clitrace\Clitrace.cli
  ```
- An existing directory that you have write permission for.

  ```
  db2trc fmt –cli c:\temp\clitrace\dump.file c:\temp
  ```

If the *CLItraceName* directory exists and you have write permission to the directory, the trace is formatted into separate text files that are based on the thread ID of a process. If there is a directory with the same name as the *CLItraceName* value and you do not have write permission to the directory, an error is returned. The name of the trace files in the *CLItraceName* directory consists of `p<pid>t<tid>.cli`.

Tracing the CLI driver by using the `db2cli.ini` file:

To turn on a CLI trace, you can enable a set of CLI configuration keywords in the `db2cli.ini` file.

About this task

You can specify the followingCLI trace keywords in the `db2cli.ini` file:

- Trace
- TraceAPIList
- TraceAPIList!
- TraceComm
- TraceErrImmediate
- TraceFileName
- TraceFlush
- TraceFlushOnError
- TraceLocks
- TracePathName
- TracePIDList
- TracePIDTID
- TraceRefreshInterval
- TraceStmtOnly
- TraceTime
- TraceTimeStamp

Procedure

To obtain a CLI trace:

1. Update the `db2cli.ini` file with CLI configuration keywords.

   You can update the `db2cli.ini` file by either manually editing the `db2cli.ini` file or by issuing the **`UPDATE CLI CFG`** command.

   - To manually edit the `db2cli.ini` file:
     a. Locate the `db2cli.ini` file. For more location of the `db2cli.ini` file, see *Call Level Interface Guide and Reference Volume 1.*
     b. Open the `db2cli.ini` file in a plain text editor.

c. Add the following section to the db2cli.ini file or, if the [COMMON] section exists, append the CLI trace keywords in the following example:

```
[COMMON]
Trace=1
TracePathName=path
TraceComm=1
TraceFlush=1
TraceTimeStamp=1
```

If you use the **TracePathName** keyword, ensure that the *path* that you specify exists and that it has global read and write permission.

Note:

– Because CLI trace keywords are in the [COMMON] section of the db2cli.ini file, their values apply to all database connections that are made through the CLI driver.

– CLI trace keywords are not case-sensitive. However, path and file name keyword values might be case-sensitive on some operating systems, such as UNIX operating systems.

– If a CLI trace keyword in the db2cli.ini file is invalid, the CLI trace utility ignores the CLI trace keyword. If you specify a valid CLI keyword with an invalid value, the default value for that trace keyword is used instead.

– Unless you set the **TraceRefreshInterval** keyword, CLI trace keywords are read from the db2cli.ini file only once, at application initialization time. If you set the **TraceRefreshInterval** keyword, the **Trace** and **TracePIDList** keywords are reread from the db2cli.ini file at the specified interval and applied, as appropriate, to the currently running application.

d. Add at least one blank line at the end of the file. Addition of a blank line prevents some parsing errors.

e. Save the file.

- To use the **UPDATE CLI CFG** command to update the CLI configuration keywords:

  a. Issue the following commands:

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

  If you use the **TracePathName** keyword, ensure that the *path* that you specify exists and that it has global read and write permission.

  b. Verify the CLI trace keywords in the db2cli.ini configuration file by issuing the following command:

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

  Note:

  – The IBM Data Server Driver Package and IBM Data Server Driver for ODBC and CLI installations do not contain the Db2 command line processor. To change the settings of trace configuration keywords, you can modify the db2cli.ini file manually.

2. To enable the CLI trace, restart the application. If you are tracing a CLI stored procedure, restart the Db2 instance

The `db2cli.ini` file is read only on application initialization, unless the **`TraceRefreshInterval`** keyword is set.

3. Capture the error:
   a. Run the application until the error is generated. To reduce the trace size, if possible, run only the application that is required to replicate the problem.
   b. Terminate the application.
4. Disable the CLI trace setting in one of the following ways:
   - Set the **`Trace`** keyword to a value of `0` in the `[COMMON]` section of the `db2cli.ini` file.
   - Issue the following command:
     ```
     db2 UPDATE CLI CFG FOR SECTION COMMON USING Trace 0
     ```
5. To disable the tracing, restart the application.

Results

The CLI trace files are written to the path that you specified for the **`TracePathName`** keyword. The file names have a format of `p`*`pid`*`t`*`tid`*`.cli`. The *pid* value is the process ID that the operating system assigns, and the *tid* value is a numeric counter (starting at 0) for each thread that is generated by the application process. An example of the file name is `p1234t1.cli`.

What to do next

The CLI trace facility has significant effect on performance of all applications currently running on the instance. Therefore, it is important to turn off the CLI trace after the required information is gathered.

If you are working with IBM Service to diagnose a problem, gather all of the files that are present in the trace path.

Interpreting input and output parameters in CLI trace files:

As is the case with any regular function, Db2 Call Level Interface (CLI) functions have input and output parameters. In a CLI trace, these input and output parameters can be seen, providing details about how each application is invoking a particular CLI API. The input and output parameters for any CLI function, as shown in the CLI trace, can be compared to the definition of that CLI function in the CLI Reference sections of the documentation.

Here is a snippet from a CLI trace file:

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

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

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

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

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

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

Analyzing Dynamic SQL in CLI traces:

CLI Traces also show how dynamic SQL is performed by 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 (a question mark (?) or a colon followed by a name (:*name*) denotes a parameter marker):

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

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

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

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

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

Interpreting timing information in CLI traces:

There are a few ways to gather timing information from a CLI trace. By default, a CLI trace captures the time spent in the application since the last CLI API call was made on a 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 CLI API call. The keyword has 4 display options: no timestamp information, processor ticks and ISO timestamp, processor ticks, or ISO timestamp.

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

Interpreting unknown values in CLI traces:

It is possible that a CLI function might return "Unknown value" as a value for an input parameter in a CLI trace.

This can occur if the 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 outdated 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 this 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 cannot be created. These return codes are not externalized and thus at this point you must contact IBM Software Support if you want further information about the meaning of the return code.

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

Interpreting multi-threaded CLI trace output:

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

If you must 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).

Analyzing CLI trace file created using the db2cli.ini file:

Applications that access the CLI driver can make use of the CLI trace utility. This utility records all function calls made by the CLI drivers to a trace file, which is useful for problem determination.

This topic discusses how to access and interpret the trace files generated by the tracing utility:
- CLI trace file location
- CLI trace file interpretation

CLI trace file location

If the **`TraceFileName`** keyword was used in the `db2cli.ini` file to specify a fully qualified file name, then the CLI trace file will be in the location specified. If a relative file name was specified for the CLI trace file name, the location of that file will depend on what the operating system considers to be the current path of the application.

Note: If the user executing the application does not have sufficient authority to write to the trace file in the specified path, no file will be generated and no warning or error is given.

The CLI trace utility automatically use the application's process ID and thread sequence number to name the trace files when the **`TracePathName`** keyword have been set. For example, a CLI trace of an application with three threads might generate the following CLI trace files: `p1680t3124.cli`, `p1680t3125.cli`, `p1680t3126.cli`.

If no CLI trace output files appear to have been created:

- Verify that the trace configuration keywords are set correctly in the `db2cli.ini` file. Issuing the `db2 GET CLI CFG FOR SECTION COMMON` command from the command line processor is a quick way to do this.
- Ensure the application is restarted after updating the `db2cli.ini` file. The CLI trace utility are initialized during application startup. The CLI trace utility can be reconfigured at run time but only if the **`TraceRefreshInterval`** keyword was appropriately specified before the application startup.

  Note: Only the **`Trace`** and **`TracePIDList`** CLI keywords can be reconfigured at run time. *Changes made to other CLI keywords, including* ***`TraceRefreshInterval`****, have no effect without an application restart*.
- If the **`TraceRefreshInterval`** keyword was specified before the application startup, and if the **`Trace`** keyword was initially set to `0`, ensure that enough time has elapsed for the CLI trace facility to reread the **`Trace`** keyword value.
- If the **`TracePathName`** keyword is used to specify trace directory ensure given directory exist before starting the application.
- Ensure the application has write access to the specified trace file or trace directory.
- Check the **`DB2CLIINIPATH`** environment variable. If set, the CLI trace facilities expect the `db2cli.ini` file to be at the location specified by this variable.
- If "Db2 Common Application Data path" or **`DB2_COMMON_APP_DATA_TOP_PATH`** response file keyword was specified during the Db2 product installation, check the path specified in **`DB2_COMMON_APP_DATA_PATH`** environment variable (only available on Windows operating systems).
- If the application uses ODBC to interface with the CLI driver, verify that one of the `SQLConnect()`, `SQLDriverConnect()` or `SQLBrowseConnect()` functions have been successfully called. No entries will be written to the CLI trace files until a database connection has successfully been made.

CLI trace file interpretation

CLI traces always begin with a header that identifies the process ID and thread ID of the application that generated the trace, the time the trace began, and product specific information such as the local Db2 build level and CLI driver version. For example:

```
1 [ Process: 1356, Thread: 5656 ]
2 [ Date & Time:              08/05/2011 15:35:00.326001 ]
3 [ Product:                  QDB2/NT Db2 v9.7.400.501 ]
4 [ Level Identifier:         08050107 ]
5 [ CLI Driver Version:       09.02.0000 ]
6 [ Informational Tokens:     "Db2 v9.7.400.501","s110330","IP23237","Fixpack 4" ]
7 [ Install Path:             C:\PROGRA~1\IBM\SQLLIB ]
8 [ db2cli.ini Location2:     C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\cfg\
db2cli.ini ]
9 [ CLI Driver Type:          IBM Data Server Runtime Client ]
```

Note: Trace examples used in this section have line numbers preceding them. These line numbers have been added to aid the discussion and will *not* appear in an actual CLI trace.

Immediately following the trace header, there are usually a number of trace entries related to environment and connection handle allocation and initialization. For example:

```
10  SQLAllocEnv( phEnv=&bffff684 )
11      ---> Time elapsed - +9.200000E-004 seconds

12  SQLAllocEnv( phEnv=0:1 )
13      <--- SQL_SUCCESS   Time elapsed - +7.500000E-004 seconds
```

```
14  SQLAllocConnect( hEnv=0:1, phDbc=&bffff680 )
15      ---> Time elapsed - +2.334000E-003 seconds

16  SQLAllocConnect( phDbc=0:1 )
17      <--- SQL_SUCCESS   Time elapsed - +5.280000E-004 seconds

18  SQLSetConnectOption( hDbc=0:1, fOption=SQL_ATTR_AUTOCOMMIT, vParam=0 )
19      ---> Time elapsed - +2.301000E-003 seconds

20  SQLSetConnectOption( )
21      <--- SQL_SUCCESS   Time elapsed - +3.150000E-004 seconds

22  SQLConnect( hDbc=0:1, szDSN="SAMPLE", cbDSN=-3, szUID="", cbUID=-3,
                          szAuthStr="", cbAuthStr=-3 )
23      ---> Time elapsed - +7.000000E-005 seconds
24  ( DBMS NAME="DB2/LINUX", Version="07.01.0000", Fixpack="0x22010105" )

25  SQLConnect( )
26      <--- SQL_SUCCESS   Time elapsed - +5.209880E-001 seconds
27  ( DSN=""SAMPLE"" )

28  ( UID=" " )

29  ( PWD="*" )
```

In the trace example, notice that there are two entries for each CLI function call (for example, lines 22-24 and 25-29 for the `SQLConnect()` function call). This is always the case in CLI traces. The first entry shows the input parameter values passed to the function call while the second entry shows the function output parameter values and return code returned to the application.

The trace example shows that the `SQLAllocEnv()` function successfully allocated an environment handle (phEnv=0:1) at line 12. That handle was then passed to the `SQLAllocConnect()` function which successfully allocated a database connection handle (phDbc=0:1) as of line 16. Next, the `SQLSetConnectOption()` function was used to set the phDbc=0:1 connection's SQL_ATTR_AUTOCOMMIT attribute to SQL_AUTOCOMMIT_OFF (vParam=0) at line 18. Finally, `SQLConnect()` was called to connect to the target database (SAMPLE) at line 22.

Included in the input trace entry of the `SQLConnect()` function on line 24 is the build and Fix Pack level of the target database server. Other information that might also appear in this trace entry includes input connection string keywords and the code pages of the client and server. For example, suppose the following information also appeared in the `SQLConnect()` trace entry:

```
( Application Codepage=819, Database  Codepage=819,
  Char Send/Recv Codepage=819, Graphic Send/Recv Codepage=819,
  Application Char Codepage=819, Application Graphic Codepage=819 )
```

This would mean the application and the database server were using the same code page (819).

The return trace entry of the `SQLConnect()` function also contains important connection information (lines 27-29 in the trace example). Additional information that might be displayed in the return entry includes any **`PATCH1`** or **`PATCH2`** keyword values that apply to the connection. For example, if `PATCH2=27,28` was specified in the `db2cli.ini` file under the `COMMON` section, the following line should also appear in the `SQLConnect()` return entry:

```
( PATCH2="27,28" )
```

Following the environment and connection related trace entries are the statement related trace entries. For example:

```
30  SQLAllocStmt( hDbc=0:1, phStmt=&bffff684 )
31      ---> Time elapsed - +1.868000E-003 seconds

32  SQLAllocStmt( phStmt=1:1 )
33      <--- SQL_SUCCESS   Time elapsed - +6.890000E-004 seconds

34  SQLExecDirect( hStmt=1:1, pszSqlStr="CREATE TABLE GREETING (MSG
                                VARCHAR(10))", cbSqlStr=-3 )
35      ---> Time elapsed - +2.863000E-003 seconds
36  ( StmtOut="CREATE TABLE GREETING (MSG VARCHAR(10))" )

37  SQLExecDirect( )
38      <--- SQL_SUCCESS   Time elapsed - +2.387800E-002 seconds
```

In the trace example, the database connection handle (phDbc=0:1) was used to allocate a statement handle (phStmt=1:1) at line 32. An unprepared SQL statement was then executed on that statement handle at line 34. If the `TraceComm=1` keyword had been set in the `db2cli.ini` file, the `SQLExecDirect()` function call trace entries would have shown additional client-server communication information as follows:

```
SQLExecDirect( hStmt=1:1, pszSqlStr="CREATE TABLE GREETING (MSG
                            VARCHAR(10))", cbSqlStr=-3 )
    ---> Time elapsed - +2.876000E-003 seconds
( StmtOut="CREATE TABLE GREETING (MSG VARCHAR(10))" )

    sqlccsend( ulBytes - 232 )
    sqlccsend( Handle - 1084869448 )
    sqlccsend( ) - rc - 0, time elapsed - +1.150000E-004
    sqlccrecv( )
    sqlccrecv( ulBytes - 163 ) - rc - 0, time elapsed - +2.243800E-002

SQLExecDirect( )
    <--- SQL_SUCCESS   Time elapsed - +2.384900E-002 seconds
```

Notice the additional sqlccsend() and sqlccrecv() function call information in this trace entry. The sqlccsend() call information reveals how much data was sent from the client to the server, how long the transmission took, and the success of that transmission (0 = SQL_SUCCESS). The sqlccrecv() call information then reveals how long the client waited for a response from the server and the amount of data included in the response.

Often, multiple statement handles will appear in the CLI trace. By paying close attention to the statement handle identifier, one can easily follow the execution path of a statement handle independent of all other statement handles appearing in the trace.

Statement execution paths appearing in theCLI trace are usually more complicated than the trace example. For example:

```
39  SQLAllocStmt( hDbc=0:1, phStmt=&bffff684 )
40      ---> Time elapsed - +1.532000E-003 seconds

41  SQLAllocStmt( phStmt=1:2 )
42      <--- SQL_SUCCESS   Time elapsed - +6.820000E-004 seconds

43  SQLPrepare( hStmt=1:2, pszSqlStr="INSERT INTO GREETING VALUES ( ? )",
                cbSqlStr=-3 )
44      ---> Time elapsed - +2.733000E-003 seconds
45  ( StmtOut="INSERT INTO GREETING VALUES ( ? )" )

46  SQLPrepare( )
```

```
47     <--- SQL_SUCCESS   Time elapsed - +9.150000E-004 seconds

48  SQLBindParameter( hStmt=1:2, iPar=1, fParamType=SQL_PARAM_INPUT,
                      fCType=SQL_C_CHAR, fSQLType=SQL_CHAR, cbColDef=14,
                      ibScale=0, rgbValue=&080eca70, cbValueMax=15,
                      pcbValue=&080eca4c )
49     ---> Time elapsed - +4.091000E-003 seconds

50  SQLBindParameter( )
51     <--- SQL_SUCCESS   Time elapsed - +6.780000E-004 seconds

52  SQLExecute( hStmt=1:2 )
53     ---> Time elapsed - +1.337000E-003 seconds
54  ( iPar=1, fCType=SQL_C_CHAR, rgbValue="Hello World!!!", pcbValue=14,
      piIndicatorPtr=14 )

55  SQLExecute( )
56     <--- SQL_ERROR   Time elapsed - +5.951000E-003 seconds
```

In the trace example, the database connection handle (phDbc=0:1) was used to allocate a second statement handle (phStmt=1:2) at line 41. An SQL statement with one parameter marker was then prepared on that statement handle at line 43. Next, an input parameter (iPar=1) of the appropriate SQL type (SQL_CHAR) was bound to the parameter marker at line 48. Finally, the statement was executed at line 52. Notice that both the contents and length of the input parameter (rgbValue="Hello World!!!", pcbValue=14) are displayed in the trace on line 54.

The `SQLExecute()` function fails at line 52. If the application calls a diagnostic CLI function like `SQLError()` to diagnose the cause of the failure, then that cause will appear in the trace. For example:

```
57  SQLError( hEnv=0:1, hDbc=0:1, hStmt=1:2, pszSqlState=&bffff680,
              pfNativeError=&bfffee78, pszErrorMsg=&bffff280,
              cbErrorMsgMax=1024, pcbErrorMsg=&bfffee76 )
58     ---> Time elapsed - +1.512000E-003 seconds

59  SQLError( pszSqlState="22001", pfNativeError=-302, pszErrorMsg="[IBM][CLI
       Driver][DB2/LINUX] SQL0302N  The value of a host variable in the EXECUTE
       or OPEN statement is too large for its corresponding use.
       SQLSTATE=22001", pcbErrorMsg=157 )
60     <--- SQL_SUCCESS   Time elapsed - +8.060000E-004 seconds
```

The error message returned at line 59 contains the Db2 native error code that was generated (SQL0302N), the sqlstate that corresponds to that code (SQLSTATE=22001) and a brief description of the error. In this example, the source of the error is evident: on line 52, the application is trying to insert a string with 14 characters into a column defined as VARCHAR(10) on line 34.

If the application does not respond to a CLI function warning or error return code by calling a diagnostic function like `SQLError()`, the warning or error message should still be written to the CLI trace. However, the location of that message in the trace may not be close to where the error actually occurred. Furthermore, the trace will indicate that the error or warning message was not retrieved by the application. For example, if not retrieved, the error message in the example might not appear until a later, seemingly unrelated CLI function call as follows:

```
SQLDisconnect( hDbc=0:1 )
    ---> Time elapsed - +1.501000E-003 seconds
    sqlccsend( ulBytes - 72 )
    sqlccsend( Handle - 1084869448 )
    sqlccsend( ) - rc - 0, time elapsed - +1.080000E-004
    sqlccrecv( )
    sqlccrecv( ulBytes - 27 ) - rc - 0, time elapsed - +1.717950E-001
```

```
( Unretrieved error message="SQL0302N  The value of a host variable in the
  EXECUTE or OPEN statement is too large for its corresponding use.
  SQLSTATE=22001" )

SQLDisconnect( )
    <--- SQL_SUCCESS   Time elapsed - +1.734130E-001 seconds
```

The final part of a CLI trace should show the application releasing the database connection and environment handles that it allocated earlier in the trace. For example:

```
SQLTransact( hEnv=0:1, hDbc=0:1, fType=SQL_ROLLBACK )
    ---> Time elapsed - +6.085000E-003 seconds
( ROLLBACK=0 )

SQLTransact( )
    <--- SQL_SUCCESS   Time elapsed - +2.220750E-001 seconds

SQLDisconnect( hDbc=0:1 )
    ---> Time elapsed - +1.511000E-003 seconds

SQLDisconnect( )
    <--- SQL_SUCCESS   Time elapsed - +1.531340E-001 seconds

SQLFreeConnect( hDbc=0:1 )
    ---> Time elapsed - +2.389000E-003 seconds

SQLFreeConnect( )
    <--- SQL_SUCCESS   Time elapsed - +3.140000E-004 seconds

SQLFreeEnv( hEnv=0:1 )
    ---> Time elapsed - +1.129000E-003 seconds

SQLFreeEnv( )
    <--- SQL_SUCCESS   Time elapsed - +2.870000E-004 seconds
```

Platform-specific tools

Diagnostic tools (Windows):

Three useful diagnostic tools on Windows systems are described.

The following diagnostic tools are available for Windows operating systems:

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

Task Manager
: The Task Manager shows all of the processes running on the Windows server, along with details about memory usage. Use this tool to find out which Db2 processes are running, and to diagnose performance problems. Using this tool, you can determine memory usage, memory limits, swapper space used, and memory leakage for a process.

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

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

Diagnostic tools (Linux and UNIX):

This section describes some essential commands for troubleshooting and performance monitoring on Linux and UNIX platforms.

For details on any one of these commands, precede it with "man" on the command line. Use these commands to gather and process data that can help identify the cause of a problem you are having with your system. Once the data is collected, it can be examined by someone who is familiar with the problem, or provided to IBM Software Support if requested.

Troubleshooting commands (AIX)

The following AIX system commands are useful for Db2 troubleshooting:

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

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

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

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

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

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

spmon
If you are using system partitioning as part of the Parallel System Support Program (PSSP), you might need to check if the SP Switch is running on all workstations. To view the status of all database partitions, use one of the following commands from the control workstation:

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

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

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

The asterisk does not display 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 seem to be suspended.

Performance Monitoring Tools

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

vmstat
This command is useful for determining if something is suspended or just taking a long time. You can monitor the paging rate, found under the page in (pi) and page out (po) columns. Other important columns are the amount of allocated virtual storage (avm) and free virtual storage (fre).

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

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

system file
Available for Solaris operating system, the `/etc/system` file contains definitions for kernel configuration limits such as the maximum number of users allowed on the system at a time, the maximum number of processes per user, and the interprocess communication (IPC) limits on size and number of resources. These limits are important because they affect Db2 performance on a Solaris operating system machine.

Troubleshooting Db2 servers

In general, the troubleshooting process requires that you isolate and identify a problem, then seek a resolution. This section will provide troubleshooting information related to specific features of Db2 products.

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

The following questions direct you to appropriate troubleshooting tasks:

1. Have you applied all known fix packs? If not, consider “Getting fixes” on page 257.
2. Does the problem occur when you are:
   - Installing Db2 database servers or clients? If so, refer to “Collecting data for installation problems” on page 145.
   - Creating, dropping, updating or upgrading an instance or the Db2 Administration Server (DAS)? If so, refer to “Collecting data for DAS and instance management problems” on page 124.
   - Moving data using **EXPORT**, **IMPORT**, **LOAD** or **db2move** commands? If so, refer to “Collecting data for data movement problems” on page 124.

If your problem does not fall into one of these categories, basic diagnostic data might still be required if you are contacting IBM Software Support. Refer to “Collecting data for Db2.”

Collecting data for Db2

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

Before you begin

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

Procedure

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

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

Using **-s** will give system details about the hardware used and the operating system. Using **-d** will give details about the specified database. Using **-c** allows for an attempt to connect to the specified database.
The output is conveniently collected and stored in a compressed ZIP archive, `db2support.zip`, so that it can be transferred and extracted easily on any system.

What to do next

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

You can do any of the following tasks next:

- Analyze the data
- Submit the data to IBM Software Support

Collecting data for data movement problems:

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

Follow the data collection instructions, appropriate for the circumstance you are experiencing, from the following list:

- To collect data for problems related to the **db2move** command, go to the directory where you issued the command. Locate the following file(s), depending on the action you specified in the command:
  - For the COPY action, look for files called `COPY.`*`timestamp`*`.ERR` and `COPYSCHEMA.`*`timestamp`*`.MSG`. If you also specified either LOAD_ONLY or DDL_AND_LOAD mode, look for a file called `LOADTABLE.`*`timestamp`*`.MSG` as well.
  - For the EXPORT action, look for a file called EXPORT.out.
  - For the IMPORT action, look for a file called IMPORT.out.
  - For the LOAD action, look for a file called LOAD.out.
- To collect data for problems related to **EXPORT**, **IMPORT**, or **LOAD** commands, determine whether your command included the MESSAGES parameter. If it did, collect the output file. These utilities use the current directory and the default drive as the destination if you do not specify otherwise.
- To collect data for problems related to a **REDISTRIBUTE** command, look for a file called "*databasename.database_partition_groupname. timestamp*" on Linux and UNIX operating systems and "*databasename. database_partition_groupname.date.time*" on Windows operating systems. It is located in the *$HOME*`/sqllib/db2dump` directory on Linux and UNIX operating systems and in the `$DB2PATH\sqllib\redist` directory on Windows operating systems, where *$HOME* is the home directory of the instance owner.

Collecting data for DAS and instance management problems:

If you are experiencing problems while performing Db2 Administration Server (DAS) or instance management and you cannot determine the cause of the problem, collect diagnostic data that either you or IBM Software Support can use to diagnose and resolve the problem.

Important: The Db2 Administration Server (DAS) has been deprecated in Version 9.7 and might be removed in a future release. The DAS is not supported in Db2 pureScale environments. Use software programs that use the Secure Shell protocol for remote administration. For more information, see "Db2 administration server (DAS) has been deprecated" at .

These steps are only for situations where you can re-create the problem and you are using Db2 on Linux or UNIX.

To collect diagnostic data for DAS or instance management problems:

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

   ```
   db2setup -t trace.out
   dascrt -u DASUSER -d
   dasdrop -d
   dasmigr -d
   dasupdt -d
   ```

```
db2icrt -d INSTNAME
db2idrop INSTNAME -d
db2iupgrade -d INSTNAME
db2iupdt -d INSTNAME
```

2. Locate the diagnostic files. More than one file might be present, so compare the timestamps to ensure that you are obtaining all of the appropriate files.

   The output will be found in the `/tmp` directory by default.

   Example file names are: `dascrt.log`, `dasdrop.log` , `dasupdt.log` , `db2icrt.log.PID`, `db2idrop.log.PID`, `db2iupgrade.log.PID`, and `db2iupdt.log.PID`, where PID is the process ID.
3. Provide the diagnostic file(s) to IBM Software Support.

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

Collecting diagnostic data for specific performance problems:

Collecting diagnostic data for performance problems that is as problem-specific as possible helps you avoid the impact of collecting the full set of performance-related diagnostic data, which is especially important on a system that is already resource constrained. The performance problems that you can collect specific diagnostic data for must be related to processor usage, memory or database connections.

Collecting diagnostic data for these problems can start once you observe that one or more of the problem symptoms exists. To collect the data required to diagnose the problem, use the **db2fodc** command together with one of the **-connections**, **-cpu**, or **-memory** parameters.

For problems that occur only sometimes, you can also create a threshold rule with the **db2fodc -detect** command that will detect a specific problem condition and start the collection of diagnostic data if the trigger conditions you specify are exceeded.

Symptoms
Depending on the specific problem that exists, you might observe some of the following symptoms on your data server:
- For performance problems related to database connections, you might observe sudden spikes in the number of application in the executing or compiling state or new database connections are being denied.
- For performance problems related to processor usage, you might observe high processor utilization rates, a high number of running processes, or high processor wait times.
- For performance problems related to memory usage, you might observe that no free memory is available, that swap space is being used at a high rate, that excessive paging is taking place, or you might suspect a memory leak.

Diagnosing the problem
To check for application connections to the database, you can issue the **db2pd -applications** command. Apart from determining how many connections to the database exist, the `Status` field can tell you if the applications are in the executing

or compiling state. If the number of applications connected to the database continues to grow over time without ever becoming lower, or if the number of application connections spikes suddenly, you might have a problem related to connections are that needs to be diagnosed further.

On UNIX and Linux operating systems, you can check whether you have a potential problem related to processor usage by issuing the **vmstat** command with the appropriate parameters for the specific operating system to return the user processor usage value (`us`). You can also check the values returned for `sy` to obtain system processor usage and the combined user and system processor usage information. You can issue the **vmstat** command several times to arrive at a more accurate conclusion about whether the problem symptoms are persisting over time. If you notice that the `us` value is frequently greater than 90% then this indicates that you might have a problem related to processor usage that needs to be diagnosed further. On Windows operating systems, you can use the **db2pd -vmstat** command to obtain similar information.

To check if you have a potential memory-related problem, you can issue the appropriate operating system command that returns the used swap space value (on Linux operating systems you can use the **free** command, for example). If the used swap space value is consistently greater than 90% then you might have a memory-related performance problem.

Resolving the problem

The following examples use the **db2fodc** command together with the **-connection**, **-cpu**, **-memory**, and **-detect** parameters. You can run these commands yourself, but you need to be aware of the potential impact: on a very resource constrained system even very problem-specific diagnostic data collection can place additional demands on the system that might not be acceptable. It is preferable to run these commands under the guidance of IBM support. Typically, the collected data is sent to IBM technical support for analysis.

- To collect diagnostic data for problems related to database connections, issue the following command:

  ```
  db2fodc -connections
  ```

  This command collects connection-related diagnostic data and places it in a `FODC_Connections_`*`timestamp_member`* directory, where *timestamp* is the time when the **db2fodc -connections** command was executed and *member* is the member or members the collection was performed for.
- To collect diagnostic data for problems related to processor usage when you are already observing related problem symptoms, you can issue the following command:

  ```
  db2fodc -cpu
  ```

  The **db2fodc -cpu** command collects processor-related diagnostic data and places it in a `FODC_Cpu_`*`timestamp_member`* directory, where *timestamp* is the time when the **db2fodc -connections** command was executed and *member* is the member or members the collection was performed for. As an alternative when the problem is intermittent or if you want to configure your system ahead of time to collect diagnostic data when a specific problem condition exists, you can issue a variation of the following command:

  ```
  db2fodc -cpu basic -detect us">=90" rqueue"<=1000" condition="and" interval="2"
   sleeptime="500" triggercount="3" iteration="4" duration="500" -member all
  ```

The **-detect** parameter with the threshold rules specified delays the collection of processor-related information until the trigger conditions specified by the threshold rule are detected. You can specify your own rules for the **-detect** parameter to determine when to start diagnostic data collection. In the previous example, conditions for user processor usage and the run queue must both be met three times over the course of three iterations. This means that the trigger conditions must exist for 6 seconds total to trigger diagnostic data collection on all members (a trigger count of 3 x 2 second intervals = a trigger condition that must exist for 6 seconds). The `iteration` option specifies that trigger condition detection followed by diagnostic data collection is performed tree times, with a sleep time of 500 seconds between each iteration.

- To collect diagnostic data for problems related to memory usage when you are already observing related problem symptoms, you can issue the following command:

  ```
  db2fodc -memory
  ```

  The **db2fodc -memory** command collects memory-related diagnostic data and places it in the `FODC_Memory_`*`timestamp_member`*, where *timestamp* is the time when the **db2fodc -connections** command was executed and *member* is the member or members the collection was performed for. As an alternative when the problem is intermittent or if you want to configure your system ahead of time to collect diagnostic data when a specific problem condition exists, you can issue a variation of the following command:

  ```
  db2fodc -memory basic -detect free"<=10" connections">=1000" sleeptime="1"
  iteration="10" interval="10" triggercount="4" duration="5" -member 3
  ```

  The **-detect** parameter with the rules specified delays collection until the rules are detected. In this example, the trigger condition for free memory and the number of connections to the database must exist for 40 seconds to trigger diagnostic data collection on member 3 (trigger count of 4 x 10 second intervals = 40 seconds total). Ten iterations of detection and diagnostic data collection can be performed, enabled over a duration of 5 hours.
- To only detect trigger conditions but not to perform diagnostic data collection, you can use the **db2fodc -detect** command together with the `nocollect` option. An entry is logged in the db2diag log files anytime the problem condition specified is detected as a threshold hit. If you choose not to collect diagnostic data, then, for detailed threshold hits, the db2diag log entry must be used to determine if a problem condition that you created a threshold rule for was detected.

  ```
  2011-05-31-15.53.42.639270-240 I2341E790 LEVEL: Event
  PID : 13279 TID : 47188095859472PROC : db2fodc
  INSTANCE: kunxu NODE : 000
  FUNCTION: Db2, RAS/PD component, pdFodcDetectAndRunCollection, probe:100
  CHANGE :

  Hostname: hotel36 Member(s): 0 Iteration: 0
  Thresholds hit 0: cs(0)>=0 us(0)<=50 rQueue(0)>=0 free(7980936)>=100
  avm(7297432)>=1 swapused(0)>=0 UOW_Executing(4)>=3
  Thresholds hit 1: cs(0)>=0 us(0)<=50 rQueue(0)>=0 free(7981000)>=100
  avm(7297528)>=1 swapused(0)>=0 UOW_Executing(4)>=3
  Thresholds hit 2: cs(0)>=0 us(0)<=50 rQueue(0)>=0 bQueue(1)>=1
  free(7981420)>=100 avm(7297596)>=1 swapused(0)>=0 UOW_Executing(4)>=3
  Thresholds hit 3: cs(0)>=0 us(0)<=50 rQueue(0)>=0 free(7981436)>=100
  avm(7297668)>=1 swapused(0)>=0 UOW_Executing(4)>=3
  ```

In this example, the problem condition specified is detected as a threshold hit four times on member 0. Both the values you specified for each threshold rule and the actual values detected are logged.

Related information

Best practices: Troubleshooting Db2 servers

Analyzing data for Db2

After you collect data, you must determine how that data can help you to resolve your particular problem. The type of analysis depends on the type of problem that you are investigating and the data that you have collected. These steps represent how to start your investigation of any basic Db2 diagnostic data.

Procedure

To analyze diagnostic data, take the following actions:

- Have a clear understanding of how the various pieces of data relate to each other. For example, if the data spans more than one system, keep your data well organized so that you know which pieces of data come from which sources.
- Confirm that each piece of diagnostic data is relevant to the timing of the problem by checking timestamps. Note that data from different sources can have different timestamp formats; be sure to understand the sequence of the different elements in each timestamp format so that you can tell when the different events occurred.
- Determine which data sources are most likely to contain information about the problem, and start your analysis there. For example, if the problem is related to installation, start your analysis with the installation log files (if any), rather than starting with the general product or operating system log files.
- The specific method of analysis is unique to each data source, but one tip that is applicable to most traces and log files is to start by identifying the point in the data where the problem occurs. After you identify that point, you can work backward in time through the data to unravel the root cause of the problem.
- If you are investigating a problem for which you have comparative data from an environment that is working and one that is not, start by comparing the operating system and product configuration details for each environment.

Recovering from sustained traps

The Db2 instance prepares the first occurrence data capture (FODC) package for the trap that you have encountered. By default, the Db2 instance has been configured for trap resiliency. The Db2 instance has also determined whether or not the trap is sustainable. The term "sustainable" means that the trapped Db2 engine thread has been suspended and the Db2 instance continues to run.

About this task

By default, the Db2 instance has been configured for trap resiliency based on the default setting of the **DB2RESILIENCE** registry variable.

Recognizing a sustained trap

Traps are sustained to minimize the effect on the database system when traps (Db2 programming errors) occur. A sustained trap results in the following diagnostics:

1. An FODC directory is created under the fully qualified path specified with the **diagpath** database manager configuration parameter.

2. Error message ADM14013C is logged to the administration notification and db2diag log files.

   Note: ADM14011C is logged if the trap could not be sustained, resulting in the instance being shut down.
3. Error sqlcode -1224 is returned to the application.
4. The EDU thread is suspended, which can be observed in the output of **db2pd -edus**.

Recovery

While it is expected that a sustained trap does not hamper the regular operation of the instance, a suspended EDU thread does hold on to some resources, and it is recommended to stop and restart the instance at your earliest convenience by following these steps:

1. To terminate all active applications which issue a COMMIT or ROLLBACK within the timeout period, which minimizes the recovery window for crash recovery when the **db2start** command is run, issue the following command:
   ```
   db2 quiesce instance instance_name user user_name
          defer with timeout minutes
   ```
2. [Optional] To terminate any applications that did not COMMIT or ROLLBACK during the timeout period in Step 1 and any new applications which accessed the database after the timeout period completed, issue the following command:
   ```
   db2 quiesce instance instance_name user user_name immediate
   ```
3. To shut down the instance, issue:
   ```
   db2stop force
   ```
4. Restart the Db2 instance using either one of the following commands:
   ```
   db2start
   ```
   or
   ```
   db2 START DATABASE MANAGER
   ```

Diagnosis

Locate the FODC directory that is specified under the **diagpath** database manager configuration parameter. The location of the FODC directory can also be confirmed by viewing the administration notification or db2diag log files. Forward the FODC information to IBM Software Support.

Identifying db2diag log entries for a load operation

Identifying diagnostic information for a load operation and finding the **db2diag** command log messages associated with that load operation are important steps in troubleshooting problematic load operations.

Symptoms

When issuing multiple load operations you might notice that a load operation is taking longer to complete than normal or appears to be hanging.

Resolving the problem

To determine which of the load operations are being problematic:

1. Issue the **db2pd -utilities** command to display the IDs of all load operations. In the following example output, there is only one load operation that is taking place, with an ID of `LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)`:

```
Database Partition 0 -- Active -- Up 0 days 00:16:15 -- Date 05/13/2011 13:00:33

Utilities:
```

```
Address             ID   Type   State   Invoker   Priority   StartTime             DBName   NumPhases   CurPhase   Description
0x000000020120E2A0  1    LOAD   0       0         0          Fri May 13 12:44:34   SAMPLE   2           2          [LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)]...

Progress:
Address             ID   PhaseNum   CompletedWork   TotalWork   StartTime             Description
0x000000020120E600  1    1          0 bytes         0 bytes     Fri May 13 12:44:34   SETUP
0x000000020120E7E0  1    2          0 rows          0 rows      Fri May 13 12:44:34   LOAD
```

2. Issue the **db2diag -g 'dataobj:=***loadID***'** command, where *loadID* is the ID of the load operation found in the previous step. This command displays all the diagnostic log messages from the **db2diag** command log related to the specified load operation. The following example shows what is displayed when this command is issued with the load operation ID identified previously:

```
$ db2diag -g 'dataobj:= LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)'


2011-05-13-14.27.03.134598-240 I1700E525           LEVEL: Warning

PID     : 29615                      TID  : 47039995963712PROC : db2sysc

INSTANCE: vivmak                     NODE : 000          DB   : SAMPLE

APPHDL  : 0-7                        APPID: *LOCAL.vivmak.110513164421

AUTHID  : VIVMAK

EDUID   : 44                         EDUNAME: db2lrid

FUNCTION: Db2, database utilities, sqlulPrintPhaseMsg, probe:314

DATA #1 : String, 99 bytes

LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)

Completed LOAD phase at 05/13/2011 14:27:03.134463.
```

After you complete these steps, you have enough information to identify the problematic load operation. However, if you need more information about the load operation, issue the **db2pd -db** *<dbname>* **-load loadID=***"LOADID"* **stacks** command to obtain a stack trace. The **stacks** option is available on UNIX and Linux operating systems only. The following example shows what is displayed when this command is issued on a sample database with the load operation ID identified previously:

```
$ db2pd -db sample -load loadID="LOADID: 16.2011-05-13-12.34.34.638811.0 (3;5)" stacks

Attempting to produce stack traces for LOAD edu 40

Attempting to produce stack traces for LOAD edu 41

Attempting to produce stack traces for LOAD edu 42

Attempting to produce stack traces for LOAD edu 44

Attempting to produce stack traces for LOAD edu 45


Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:27 -- Date 05/13/2011 14:28:32

Node Number : 0

Database Name : SAMPLE

LoadID                                          EDUID  EDUNAME   TableName  SchemaName  AppHandl [nod-index]  Application ID               StartTime                   LoadPhase

LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  40     db2lfrm0  T1         VIVMAK      7        [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD

LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  41     db2lfrm1  T1         VIVMAK      7        [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD

LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  42     db2lfrm2  T1         VIVMAK      7        [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD

LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  44     db2lrid   T1         VIVMAK      7        [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD

LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)  45     db2lbm0   T1         VIVMAK      7        [000-00007]  *LOCAL.vivmak.110513164421  2011-05-13-12.44.34.638811  LOAD
```

The **db2pd -db** *<dbname>* **-load loadID=***"LOADID"* **stacks** command displays all the EDU information related to the load operation specified and produces stack trace files in the `diagpath` directory.

You can use all the information retrieved to perform further troubleshooting techniques, such as monitoring or terminating the load operation. Also, the information gathered might be requested by IBM technical support to troubleshoot problematic load operations.

You can also use the collected information to run the **db2trc** command for further troubleshooting. To run the **db2trc** command for a specific load operation by using the retrieved information:

1. Run the **db2pd -load** command to retrieve the application ID of the specific load operation that you are interested in.
2. Run the **db2trc -appid** or **db2trc -apphdl** command to record further information about the load operation.

In the following example, the application ID `*LOCAL.vivmak.110513164421` of the load ID `LOADID: 16.2011-05-13-12.44.34.638811.0 (3;5)` from the previous example in this topic, is used to run the **db2trc** command:

```
$ db2trc  on -appid  *LOCAL.vivmak.110513164421

 Trace is turned on

 $ db2trc info

 Marker                  :  @TRACE@

 Trace version           :      7.0

 Platform                :  Linux/X

 Build level             :  n110612

 maxBufferSize           : 33554432 bytes (32 MB)

 auxBufferSize           : 524288 bytes (0 MB)

 allocationCount         : 2

 DB2TRCD pid             : 0

 Trace destination       : <shared memory buffer>

 numSuspended            : 0

 Trace starting time     : 2011-06-16-10.07.52.209999-240


 Buffer size             : 33554432 bytes (32 MB)

 Allow buffer to wrap    : yes

 Mask                    : *.*.*.*.*

 Timestamps              : disabled

 PID.TID mask            : all

 Fixed data mask #1      : all

 Fixed data mask #2      : all

 Max system errors       : infinite

 Treat this rc as sys err: none
 Member mask             : none

 Application handle mask : none

 Application ID mask     : *LOCAL.vivmak.110513164421
```

In the next example, the application handle obtained from the output of the **db2pd -load** command is used to change the trace options that are in effect for the **db2trc** command:

```
$ db2trc chg -apphdl 7

 Trace has been changed

 $ db2trc info

 Marker                  :  @TRACE@

 Trace version           :      7.0

 Platform                :  Linux/X

 Build level             :  n110612

 maxBufferSize           : 33554432 bytes (32 MB)

 auxBufferSize           : 524288 bytes (0 MB)

 allocationCount         : 2

 DB2TRCD pid             : 0

 Trace destination       : <shared memory buffer>

 numSuspended            : 0

 Trace starting time     : 2011-06-16-10.10.11.816264-240


 Buffer size             : 33554432 bytes (32 MB)

 Allow buffer to wrap    : yes

 Mask                    : *.*.*.*.*

 Timestamps              : disabled

 PID.TID mask            : all

 Fixed data mask #1      : all

 Fixed data mask #2      : all

 Max system errors       : infinite

 Treat this rc as sys err: none

 Member mask             : none

 Application handle mask : 7

 Application ID mask     : none
```

Troubleshooting administrative task scheduler

This checklist can help you troubleshoot problems that occur while running tasks in the administrative task scheduler.

Procedure

1. If your task does not execute as expected, the first thing you should do is look for a execution status record in the ADMIN_TASK_STATUS administrative view.
   - If there is a record, examine the various values. In particular, pay attention to the STATUS, INVOCATION, SQLCODE, SQLSTATE, SQLERRMC and RC columns. The values often identify the root cause of the problem.
   - If there is no execution status record in the view, the task did not execute. There are a number of possible explanations for this:
     - The administrative task scheduler is disabled. Tasks will not execute if the administrative task scheduler is disabled. To enable the scheduler, set the **DB2_ATS_ENABLE** registry variable.
     - The task was removed. Someone may have removed the task. Confirm the task exists by querying the ADMIN_TASK_LIST administrative view.
     - The scheduler is unaware of the task. The administrative task scheduler looks for new and updated tasks by connecting to each active database every five minutes. Until this period has elapsed, the scheduler is unaware of your task. Wait at least five minutes.
     - The database is inactive. The administrative task scheduler cannot retrieve or execute tasks unless the database is active. Activate the database.
     - The transaction is uncommitted. The administrative task scheduler ignores uncommitted tasks. Be sure to commit after adding, updating, or removing a task.
     - The schedule is invalid. The task's schedule might prevent the task from running. For example, the task may have already reached the maximum number of invocations. Review the task's schedule in the ADMIN_TASK_LIST view and update the schedule if necessary.
2. If you cannot determine the cause of the problem by referring to the ADMIN_TASK_STATUS administrative view, refer to the Db2 diagnostic log. All critical errors are logged to the **db2diag** log file. Informational event messages are also logged by the administrative task scheduler daemon during task execution. These errors and messages are by identified by the "Administrative Task Scheduler" component.

What to do next

If you follow the preceding steps and are still unable to determine the cause of the problem, consider opening a problem management record (PMR) with IBM Software Support. Inform them that you have followed these instructions and send them the diagnostic data that you collected.

Operation fails because database is currently in use

Some database operations such as dropping a database, terminating a database connection, and creating a backup copy of a database require the associated database to be inactive and the database can have no applications connected to it. When you attempt such an operation with an active database that has applications connected to it, the operation fails and an error with SQL code -1035N might be returned.

Symptoms

When you attempt certain database operations such as dropping a database, terminating a database connection, and creating a backup copy of a database, the operation fails and an error stating that the database is currently in use might be returned.

Causes

This SQL1035N error message might be returned in one of the following scenarios:

1. There are open connections to the database preventing the attempted operation from succeeding. This can occur in the following situations:
   - Exclusive use was requested, but the database is already in use as a shared database by another user (in the same process).
   - Exclusive use was requested, but the database is already in use as an exclusive database. This means that two different processes are trying to access the same database.
   - The maximum number of connections to the database has been reached.
   - The database is being used by another user on another system.
2. The database has been activated explicitly, preventing the operation from succeeding.
3. The database is active because it is in the WRITE SUSPEND state.

Resolving the problem

1. Take one of the following actions:
   - Resubmit the command later when the database is not in use.
   - Change the authorization to match the current user or wait until the database is not in use.
   - Wait until the database is not in exclusive use.
   - Wait until other users on another system have disconnected from the database.
   - Issue the **`QUIESCE DATABASE DEFER WITH TIMEOUT`** <*minutes*> command and a **`CONNECT RESET`** command to free connections to the database and prevent application users from reconnecting. The `DEFER` option will wait for applications until they commit the current unit of work instead of rolling back all running transactions.
   - Issue the **`LIST APPLICATIONS`** command to list connections to the database. Issue a **`FORCE APPLICATION ALL`** command to free connections to the database.

     Note: The **`FORCE APPLICATION ALL`** command is an asynchronous command which might return as successful even though it is still continuing to clean up connections. A varying interval of time might be required for the command to complete.
2. Deactivate the database using the **`DEACTIVATE DATABASE`** <*database-alias*> command.
3. Issue a **`SET WRITE RESUME FOR DATABASE`** command to resume write operations for the database.
4. Restart the member where the offline operation was taking place by issuing the following command:
   ```
   db2start member <member-number>
   ```
5. Reissue the operation that had initially failed.

Troubleshooting compression

Data compression dictionary is not automatically created:

You have a large table or a large XML storage object for the table, but the data compression dictionary was not created. You would like to understand why the creation of the data compression dictionary did not occur as you were expecting. This information applies to both the compression dictionary for the table object and the compression dictionary for the XML storage object.

You may find yourself in the following situation:

- You have a table where the COMPRESS attribute has been set to YES.
- The table has existed for some time and data has been added and removed.
- The size of the table appears to be close to the threshold size. You are expecting the data compression dictionary to be automatically created.
- You run a table data population operation (such as INSERT, LOAD INSERT, or REDISTRIBUTE) which you expect will increase the size of the table beyond the threshold size.
- Automatic creation of the data compression dictionary does not occur. The data compression dictionary is not created and placed into the table. You expect compression to occur on data added to the table after that point, but the data remains decompressed.
- For XML data, the data is in the Db2 Version 9.7 storage format.

  Compression of data in the XML storage object of a table is not supported if the table contains XML columns that were created using Db2 Version 9.5 or earlier. If you enable such a table for data row compression, only the table row data in the table object is compressed. If the XML storage object cannot be compressed during an insert, load, or reorg operation, a message is written to a db2diag log file only if the XML columns were created with Db2 V9 or Db2 V9.5.

Why is the data compression not occurring?

Although the data is larger than the threshold size to enable automatic creation of the compression dictionary, there is another condition that is checked. The condition is that there must be sufficient data present in the object to be able to create the dictionary, and message ADM5591W will inform you of this requirement. Past activity against the data may also have included the deletion or removal of data. There may be large sections within the object where there is no data. This is how you can have a large object which meets or exceeds the object size threshold, but there may not be enough data in the object to enable the creation of the dictionary.

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

If the object is sparsely populated, a reorganization of the table can be performed using the **REORG TABLE** command (use the LONGLOBDATA option for XDA) to create the dictionary. By default, KEEPDICTIONARY is specified. RESETDICTIONARY may be specified to force dictionary creation.

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

Automatic Dictionary Creation (ADC) will not occur for a table when the table is not enabled for data row compression. Message ADM5594I is returned when ADC processing is disabled for the database and describes the reason for it.

If the table contains XML columns that were created using Db2 Version 9.5 or earlier, use the ADMIN_MOVE_TABLE stored procedure to upgrade the table and then enable data row compression.

Row compression not reducing disk storage space for temporary tables:

There are known situations that might occur which result in a lack of expected savings in disk storage space for temporary tables even though the Storage Optimization feature is licensed.

Symptoms

Potential disk space savings by enabling row compression on temporary tables are not being realized as expected.

Causes

- This situation occurs mostly as a result of a large number of applications running at the same time and creating temporary tables, each of which consumes a portion of the database manager memory. This results in not enough memory being available to create the compression dictionary. Notification is not given when this situation occurs.
- Rows are compressed using a dictionary-based approach according to an algorithm. If a row of a temporary table is large enough to yield appreciable savings in disk space, the row will be compressed. Small rows in temporary tables will not be compressed and this will account for the lack of expected savings in disk storage space. Notification is not given when this situation occurs.

Risk

There is no risk to the system aside from row compression not being used on temporary tables with row sizes that do not meet the threshold. There could be other adverse effects to the database manager if available memory remains highly constricted.

Data replication process cannot decompress a compressed row image:

There are known situations that may occur which result in a data replication solution being unable to successfully decompress a log record with a compressed row image. For transient (temporary) errors, the SQL code returned will correspond to the cause of the error, while a permanent error is typically signalled by a SQL0204N notification. Only transient error situations might result in a subsequent successful decompression of a row image in a log record. The db2ReadLog API will continue processing other log records even if it cannot decompress a log record.

Symptoms

It is possible that the log reader may encounter transient and permanent errors while reading log records that contain compressed user data. Here are non-exhaustive example lists of the two classes of errors that may be encountered while reading log records with compressed data (row images).

Transient errors:

- Table space access not allowed
- Unable to access the table (lock timeout)
- Out of memory (to load and store the required dictionary)

Permanent errors:

- Table space in which the table resides does not exist
- Table or table partition to which the log record belongs does not exist
- A dictionary does not exist for the table or table partition
- The log record contains row images compressed with a dictionary older than the dictionaries in the table

Causes

It is possible that a replication solution, or any other log reader, may fall behind database activities and receive an error reading a log record which contains compressed user data (see Scenario 1). Such a case could arise if the log record being read contains compressed user data that was compressed by an older compression dictionary than what is available in the table (at the time of the log read).

Similarly, if a table is dropped, the dictionaries associated with the table will also be removed. Compressed row images for the table cannot be decompressed in this case (see Scenario 2). Note that this restriction does not apply to row images that are not in a compressed state, as these row images can still be read and replicated even if the table is dropped.

For any one table, there can be only one active data compression dictionary and one historical dictionary.

Scenario 1:

Table t6 has compression enabled. The DATA CAPTURE CHANGES attribute, for replication purposes, is enabled for the table. The table is being replicated by a data replication application and the log reader is reading log records that contain compressed data (row images). A client log reader, using the db2ReadLog API, is reading the first log record for the first INSERT statement as a **LOAD** operation is performed on table t6, after a REORG TABLE command has been issued (causing the table's dictionary to be rebuilt).

The following statements are executed against table t6, which already contains a compression dictionary and has the DATA CAPTURE CHANGES attribute is enabled:

```
-> db2 alter table t6 data capture changes
-> db2 insert into t6 values (...)
-> db2 insert into t6 values (...)
```

Since a data compression dictionary already exists for table t6, the two INSERTs after the ALTER will be compressed (using Table t6's compression dictionary). At this point, the log reader has not yet reached the first INSERT statement.

The following **REORG TABLE** command causes a new compression dictionary to be built for table t6, and the current compression dictionary is kept as the historical dictionary, thus making the log reader one dictionary behind the current compression dictionary (however, the historical dictionary is not loaded into memory after the REORG):

```
-> db2 reorg table t6 resetdictionary
```

As the log reader is reading the INSERT log for the INSERT statements, which now requires the historical dictionary to be read in memory, the table t6 is undergoing a **LOAD** operation:

```
-> db2 load from data.del of del insert into table t6 allow no access
```

When the LOAD is executed on the source table, table t6 will be Z-locked due to the specified `ALLOW NO ACCESS` option. The log reader must load the historical dictionary into memory to decompress row images found in the INSERT log records, however, fetching the dictionary requires an IN table lock. In this case, the log reader will fail to acquire the lock. This results in the sqlcode member of the db2ReadLogFilterData structure to return SQL code SQL2048N. This corresponds to a transient error (that is, the log record *might* be decompressed if the API is called again). The log reader will return the compressed row image in the log record and continue on reading the next log record.

Scenario 2:

Table t7 has the DATA CAPTURE CHANGES attribute enabled. Compression is enabled for the table in order to reduce storage costs. The table is being replicated by a data replication application, however, the log reader has fallen behind on the source table activity and the data compression dictionary has already been rebuilt twice before the log reader reads from the log records again.

The following statements are executed against Table t7, with the DATA CAPTURE CHANGES attribute already enabled, table compression is enabled, and a new dictionary is built:

```
-> db2 alter table t7 compress yes
-> db2 reorg table t7 resetdictionary
-> db2 insert into t7 values (...)
```

A client log reader, using the db2ReadLog API, is about to read the next log corresponding to the first INSERT statement in the following statements:

```
-> db2 insert into t7 values (...)
...
-> db2 reorg table t7 resetdictionary
-> db2 insert into t7 values (...)
...
-> db2 reorg table t7 resetdictionary
```

The db2ReadLog API will not be able to decompress the contents of the log record in this case, because the log reader has fallen behind two or more **REORG RESETDICTIONARY** operations. The dictionary required to decompress the row image in the log record would not be found in the table; only the compression dictionary of the second REORG and the compression dictionary of the last REORG is stored with the table. However, the db2ReadLog API would not fail with an error.

Instead, the uncompressed row image will be returned in the user buffer, and, in the db2ReadLogFilterData structure preceding the log record, the sqlcode member will return SQL code SQL0204N. This code corresponds to a permanent error (that is, the log record cannot ever be decompressed).

Environment

This failure to successfully decompress a compressed log record, due to a missing old compression dictionary, can occur on any platform on which a data replication solution uses the db2ReadLog API and the DATA CAPTURE CHANGES attribute is set for the table.

Resolving the problem

For transient errors, it may be possible to reissue the read request and successfully read the log. For example, if the log record belongs to a table residing in a table space and access to the table is not allowed, the dictionary may not be accessible to decompress the log record (see Scenario 1). The table space may become available at a later time, and reissuing the log read request at that time may successfully decompress the log record.

- If a transient error is returned (see Scenario 1), read the error information in order to take appropriate action. This may include waiting for the table operation to complete, which could allow a re-read of the log record and decompression to be successful.
- If a permanent error occurs (see Scenario 2), the row image in the log record cannot be decompressed since the compression dictionary, which was used to compress the row image, is no longer available. For this case, replication solutions may need to re-initialize the affected (target) table.

Troubleshooting global variables

Troubleshooting global variable problems:

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

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

Scenario 1

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

Scenario 2

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

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

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

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

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

db2 "connect to sample user developerUser using xxxxxxxx"

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

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

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

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

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

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

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

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

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

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

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

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

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

db2 "set session authorization = developerUser"

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

db2 "terminate"
```

Troubleshooting HADR

This troubleshooting section contains information that will help you understand and resolve problems that are specific to HADR.

Automatic HADR congestion detection and data collection:

You can use the **db2fodc** command to monitor HADR congestion. If congestion is detected, HADR diagnostic data collection is triggered automatically to aid in the problem determination process.

To monitor HADR congestion and invoke automatic diagnostic data collection, specify the **-hadr** parameter with the -detect option for the **db2fodc** command, as shown:

```
db2fodc -hadr -db dbname -detect
```

The command starts a process which monitors the HADR database to see if there is enough congestion to invoke data collection. Once there is enough congestion, the **db2cos_hadr** (**db2doc_hadr.bat** on Windows operating systems) script is invoked and diagnostic data collection occurs. The process ends once diagnostic data collection completes. If there is not enough congestion ever detected, the monitor runs until the detection duration exceeds the **duration** parameter value, or the user terminates it by issuing the following command:

```
db2fodc -detect off
```

When monitoring HADR congestion without specifying any **-detect** suboptions, default settings are being applied. The settings are as follows:

Run HADR congestion detection for 1 iteration (**iteration=1**). During this iteration, run db2pd -hadr -db *dbname* every 30 seconds (**interval=30**). For each run of the command, check the output to see whether the **HADR_CONNECT_STATUS** output is *CONGESTED*. If there are 10 (**triggercount=10**) consecutive *CONGESTED* statuses, there is enough HADR congestion to invoke the **db2cos_hadr** script. The script will invoke to collect HADR data. Once the data collection is completed, the monitoring instance will terminate. If 10 consecutive *CONGESTED* statuses do not appear, the detect iteration will run until the detect duration exceeds the **duration** parameter value, or you stop the detect by issuing the db2fodc -detect off command in another session window. Therefore, running the following two commands would be equivalent:

```
db2fodc -hadr -db dbname -detect
db2fodc -hadr -db dbname -detect iteration=1 sleeptime=0 triggercount=10 interval=30
```

Manual HADR data collection:

If you suspect that there is congestion within HADR, you can manually collect HADR related data by issuing the **db2fodc** command with the **-hadr** parameter for use in troubleshooting HADR.

To manually collect HADR related data, run the following command on the primary or standby host:

```
db2fodc -hadr -db dbname
```

This command invokes the **db2cos_hadr** (**db2cos_hadr.bat** on Windows) script and places the collected data in the FODC_Hadr_*timestamp_hostname_Primary*|*Standby*|*Standard* directory, which is created in the DIAGPATH directory.

The FODC_Hadr_*timestamp_hostname_Primary, Standby, or Standard* directory is created in the DIAGPATH directory. You can modify the **db2cos_hadr** script and place it in the sqllib/adm/directory directory so that this script is used when HADR data collection is started.

You can collect data on remote hosts by specifying the **-host** parameter. The following is an example of how to collect data on remote hosts with two hosts: hostA as primary and hostB as standby.

On hostA, issue db2fodc -hadr -db sample -host all. This command collects data on both hosts and places the collected data in the DIAGPATH directory on each host. You can also issue db2fodc -hadr -db sample -host hostA,hostB to specify HADR data collection on hostA and hostB. On hostA, issue db2fodc -hadr -db sample -host hostB to collect data on hostB and place the collected data in the DIAGPATH directory on hostB. On hostA, issue db2fodc -hadr -db sample -host all -fodcpath /TMP/hadrdata/ to collect data on both hosts and place the collected data in the /TMP/hadrdata/ directory.

Troubleshooting inconsistencies

Troubleshooting data inconsistencies:

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

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

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

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

If the **INSPECT** command does not finish, then contact IBM Software Support.

Troubleshooting index to data inconsistencies:

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

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

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

Locking implications

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

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

Troubleshooting a database in an inconsistent state:

When attempting to perform database operations, you might receive an error message stating that the database is in an inconsistent state.

Symptoms
When attempting to perform database operations, you might receive an error message stating that the database is in an inconsistent state.

Causes
The possible causes of this are:

- The database is offline as a result of an abnormal termination of the previous session (for example, a power failure).
- If the error was encountered when issuing the **db2ckupgrade** command:

  - The database is online and SQL has been issued which modified data in the database.
  - The database is online and HADR has been enabled.
- In Db2 pureScale environments only, possible causes also include:
  - The database on this Db2 member is offline as a result of an abnormal termination of the previous session.
  - The database is offline across the entire Db2 pureScale instance as a result of an abnormal termination of the previous session.
  - If drop operations are done in the instance, recoverable databases are already put into backup pending state. Drop operations are not allowed until backup of database is done.
  - An attempt was made to modify the cluster topology (for example, adding or deleting a member) while the database was in one of the following states: backup pending, restore pending, rollforward pending.

Resolving the problem

- If the database is offline as a result of an abnormal termination of the previous session, respond with the following actions
  1. Restart the database using the **RESTART DATABASE** command. In a partitioned database server environment, the command must be issued on all database partitions.
- If this error was encountered when issuing the **db2ckupgrade** command, respond with the following actions:
  1. Perform a clean shutdown of the database.
  2. After shutdown, if HADR has been enabled on the database, issue the **STOP HADR** command on the database.
  3. Reissue the **db2ckupgrade** command.
- In a Db2 pureScale environment only:
  1. If the database on this Db2 member is offline as a result of an abnormal termination of the previous session, respond with the following actions:
     a. By default, member crash recovery is automatically initiated in a Db2 pureScale environment, so no user action is required. If member crash recovery is not automatically enabled, perform member crash recovery on this Db2 member by issuing the **RESTART DATABASE** command.
     b. Some database operations can still complete on other, consistent members even if one member is inconsistent. To access the database during member crash recovery, connect to an active Db2 member. To access this specific member, wait for member crash recovery to complete.
  2. If the database is offline across the entire Db2 pureScale instance as a result of an abnormal termination of the previous session, warn users that the database will not be available until recovery completes. The next step depends on whether group crash recovery is automatically enabled. If it is enabled (the default), then no user action is required. If automatic group crash recovery is not enabled, respond with the following actions:
     a. Perform group crash recovery by issuing the **RESTART DATABASE** command.
     b. After the recovery completes, perform member crash recovery on any other members that have indoubt transactions.
  3. Back up databases that are in backup pending state, and resubmit drop operation.
  4. If the database is in a state that is not compatible with a topology change, then address the issue and then attempt the topology change request again.

Troubleshooting installation of Db2 database systems

If problems occur while you are installing Db2 database products, confirm that your system meets the installation requirements and review the list of common installation issues.

Procedure

To troubleshoot installation problems for Db2 database systems:

- Ensure that your system meets all of the installation requirements.
- If you are encountering licensing errors, ensure that you have applied the appropriate licenses.

  Review the list of frequently asked questions in the “Knowledge Collection: Db2 license issues” technote: http://www.ibm.com/support/docview.wss?rs=71&uid=swg21322757
- Review the list of installation issues in the documentation and on the Db2 Technical Support website: www.ibm.com/software/data/db2/support/db2_9/troubleshoot.html

What to do next

If you complete these steps but cannot yet identify the source of the problem, begin collecting diagnostic data to obtain more information.

Collecting data for installation problems:

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

To collect diagnostic data for installation problems:

1. Optional: Repeat the installation attempt with tracing enabled. For example:

   On Linux and UNIX operating systems
   ```
   db2setup -t /filepath/trace.out
   ```
   On Windows operating systems
   ```
   setup -t \filepath\trace.out
   ```
2. Locate the installation log files.
   - On Windows, the default file name is "`Db2-`*`ProductAbbreviation-DateTime`*`.log`". For example: `DB2-ESE-Wed Jun 21 11_59_37 2006.log`. The default location for the installation log is the `"My Documents"\DB2LOG\` directory.

     Note: If Db2 is installed using SYSTEM user, for example using Microsoft Systems Center Configuration Manager (SCCM) or Tivoli®, an install log is not created. The SYSTEM user does not have a `"My Documents"` folder for the install log to be created in. In order to view the install log, use the **`-l`** option of the setup command to create the log file in a different location.
   - On Linux and UNIX, the default file names are `db2setup.log`, `db2setup.his`, and `db2setup.err`.

     If you recreated the problem with tracing (or debug mode) enabled, additional files might be created, such as: `dascrt.log`, `dasdrop.log`, `dasupdt.log`, `db2icrt.log.`*`PID`*, `db2idrop.log.`*`PID`*, `db2iupgrade.log.`*`PID`*, and `db2iupdt.log.`*`PID`*, where *PID* is the process ID.

The default location for all of these files is the `/tmp` directory. For the trace file (trace.out), there is no default directory if not given, so you must specify the file path to the folder in which the trace output file was created.

3. Optional: If you intend to submit the data to IBM Software Support, collect data for Db2 as well. See "Collecting data for Db2" for additional information.

Analyzing data for installation problems:

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

Before you begin

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

Procedure

1. Ensure that you are looking at the appropriate installation log file. Check the file's creation date, or the timestamp included in the file name (on Windows operating systems).
2. Determine whether the installation completed successfully.
   - On Windows operating systems, success is indicated by a message similar to the following at the bottom of the installation log file:
     ```
     Property(C): INSTALL_RESULT = Setup Complete Successfully
     === Logging stopped: 6/21/2006  16:03:09 ===
     MSI (c) (34:38) [16:03:09:109]:
     Product: Db2 Enterprise Server Edition - DB2COPY1 -- Installation operation
     completed successfully.
     ```
   - On Linux and UNIX operating systems, success is indicated by a message at the bottom of the installation log file (the one named `db2setup.log` by default).
3. OPTIONAL: Determine whether any errors occurred. If the installation completed successfully, but you received an error message during the installation process, locate these errors in the installation log file.
   - On Windows operating systems, most errors will be prefaced with "`ERROR:`" or "`WARNING:`". For example:
     ```
     1: ERROR:An error occurred while running the command
     "D:\IBM\SQLLIB\bin\db2.exe
     CREATE TOOLS CATALOG SYSTOOLS USE EXISTING DATABASE TOOLSDB FORCE" to
     initialize and/or migrate the Db2 tools catalog database.
     The return value is "4".

     1: WARNING:A minor error occurred while installing "Db2 Enterprise Server
     Edition - DB2COPY1" on this computer. Some features may not function
     correctly.
     ```
   - On Linux and UNIX operating systems, a file with a default name of `db2setup.err` will be present if any errors were returned by Java (for example, exceptions and trap information).

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

Results

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

What to do next

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

Gathering diagnostic information for instance creation problems:

If you encounter problems when creating or upgrading an instance, you might need to collect diagnostic information to help troubleshoot the problem. The **`db2fodc -clp`** command helps to quickly gather the necessary diagnostic information for IBM support to analyze and troubleshoot the problem.

Symptoms
While creating or upgrading an instance, you might receive an error when trying to update the DBM CFG as part of instance creation. The error code that you might receive is DBI1281E. However, this error might not give the root cause of the problem and diagnostic information is need to further troubleshoot the instance creation problem.

Resolving the problem
To quickly gather the diagnostic information needed to troubleshoot the problem, under the guidance of IBM support, you need to perform the following steps:

1. Issue the **`db2fodc -clp`** command. This command collects environment and configuration related information that is targeted to diagnosing an instance creation problem. After collection is completed the information is stored in a newly created directory named `FODC_Clp_<`*`timestamp`*`>_<`*`member`*`>`.

After collection is completed you can send the `FODC_Clp_<`*`timestamp`*`>_<`*`member`*`>` directory to IBM support for analysis.

Known problems and solutions:

Errors when installing a Db2 database product as a non-root user to the default path on a system WPAR (AIX):

Various errors can occur if you install Db2 database products as a non-root user in the default installation path (`/opt/IBM/db2/V9.7`) on a system workload partition (WPAR) on AIX 6.1. To avoid these problems, install Db2 database products on a file system that is accessible only to the WPAR.

Symptoms

If you install Db2 database products in the `/usr` or `/opt` directories on a system WPAR, various errors can occur depending on how you configured the directories. System WPARs can be configured to either share the `/usr` and `/opt` directories with the global environment (in which case the `/usr` and `/opt` directories will be readable but not write accessible from the WPAR) or to have a local copy of the `/usr` and `/opt` directories.

In the first scenario, if a Db2 database product is installed to the default path on the global environment, that installation will be visible in the system WPAR. This will give the appearance that Db2 is installed on the WPAR, however attempts to create a Db2 instance will result in this error: `DBI1288E The execution of the program db2icrt failed. This program failed because you do not have write permission on the directory or file /opt/IBM/db2/V9.7/profiles.reg,/opt/IBM/db2/V9.7/default.env.`

In the second scenario, if a Db2 database product is installed to the default path on the global environment then when the WPAR creates the local copy of the `/usr` and `/opt` directories the Db2 database product installation will also be copied. This can cause unexpected problems if a system administrator attempts to use the database system. Since the Db2 database product was intended for another system, inaccurate information might be copied over. For example, any Db2 instances originally created on the global environment will appear to be present in the WPAR. This can cause confusion for the system administrator with respect to which instances are actually installed on the system.

Causes

These problems are caused by installing Db2 database products in `/usr` or `/opt` directories on a system WPAR.

Resolving the problem

Do not install Db2 database products in the default path on the global environment.

Mount a file system that is accessible only to the WPAR and install the Db2 database product on that file system.

Beta and non-beta versions of Db2 database products cannot coexist:

A Db2 copy can contain one or more different Db2 database products, but it cannot contain both beta and non-beta products. Do not install beta and non-beta versions of Db2 database products in the same computer.

This restriction applies to both client and server components of Db2 database products.

Resolving the problem

Uninstall the Db2 beta version before installing the non-beta version or else choose a different installation path.

Resolving service name errors when you install Db2 database products:

If you choose a non-default service name or port number for the Db2 database product or Db2 Information Center to use, ensure that you do not specify values that are already in use.

Symptoms

When you attempt to install a Db2 database product or the *Db2 Information Center*, the Db2 Setup wizard reports an error that states "The service name specified is in use".

Causes

The Db2 Setup wizard will prompt you to choose port numbers and service names when you install:
- The *Db2 Information Center*
- A Db2 database product that will accept TCP/IP communications from clients
- A Db2 database product that will act as a database partition server

This error can occur if you choose a service name and port number rather than accepting the default values. If you choose a service name that already exists in the `services` file on the system and you only change the port number, this error will occur.

Resolving the problem

Take one of the following actions:
- Use the default values.
- Use a service name and port number that are both already in the `services` file.
- Add an unused service name and an unused port number to the `services` file. Specify these values in the Db2 Setup wizard.

Errors when running the db2start command:

Errors can occur if you run the db2start command after successful installation of the Db2 database products on Windows systems. To avoid these problems, download and install the recent VC++ redistributable packages.

Symptoms

On Windows systems, if you run the **`db2start`** command after installing the Db2 database product, you might encounter the error message `Db2: The service has returned a service specific error code. SQL1042C An unexpected system error occurred. SQLSTATE=580004`.

Causes

A possible root cause of this error message is that the system has an incompatible VC++ 2008 redistributable package already installed, prior to installation of the Db2 database product.

Resolving the problem

Download and install the Microsoft VC++ 2008 redistributable packages which is available for download from http://www.microsoft.com.

Full information about the package, including download instructions is available on the Microsoft Visual C++ 2008 Redistributable Package (x86) and Microsoft Visual C++ 2008 Redistributable Package (x64) web pages.

Errors when running the db2_deinstall command in a partitioned database environment set up:

Errors can occur if you run the db2_deinstall command after successful db2idrop operation of a partitioned database environment instance.

Symptoms

`db2_deinstall` may fail with the error message `The deinstallation process cannot continue while there are Db2 instances related to the current Db2 copy. If you need to move up or down a level from this Db2 level, use the installFixPack command from the Db2 image for the level you desire to move to, which will update this copy. If you are only trying to uninstall this Db2 copy, you must either drop the related instance(s) <instance name> first, or you can update the instances to another Db2 copy which is at the same version as this Db2 copy, then restart the db2_deinstall command.`

Causes

A possible cause of the issue is that the cleanup of the Db2 instance(s) profile in the registry was either incomplete or was interrupted during the db2idrop processing. This registry entry is looked up by **`db2_deinstall`** before the uninstallation.

Resolving the problem

You must update the Db2 registry for the current instance manually using the **`db2iset`** command, including the full path of the **`db2iset`** command. For example:

```
/opt/ibm/db2/<release>/instance/db2iset -d <instance name>
```

Troubleshooting license issues

Analyzing Db2 license compliance reports:

To verify the license compliance of your Db2 offerings, analyze a Db2 license compliance report.

Before you begin

The following steps assume that you have used the **`db2licm`** command to generate a Db2 license compliance report.

Procedure

1. Open the file that contains the Db2 license compliance report.
2. Examine the status of each Db2 offering in the compliance report. The report displays one of the following values for each offering:

   In compliance
   Indicates that no violations were detected. The offering was used and is properly licensed.

   Not used
   Indicates that you did not performed any activities that require this particular offering.

   Violation
   Indicates that the offering is not licensed and was used.
3. If there are any violations, use the **`db2licm -l`** command to view your license information.

   If the Db2 offering is listed with a status of `"Not licensed"`, you must obtain a license for that offering. The license certificate file and instructions for registering it are available on the Activation CD that you receive when you purchase a Db2 offering.

Note: On Db2 Workgroup Server Edition the SAMPLE database includes materialized query tables (MQT), and multidimensional cluster tables (MDC) that causes a license violation. This violation can only be removed by upgrading to Db2 Enterprise Server Edition.

4. Use the following commands to determine which objects or settings in your Db2 database product are causing the license violations:
   - For the Db2 Advanced Access Control Feature:

     Check for tables that use label based access control (LBAC). Run the following command against every database in every instance in the Db2 copy:

     ```
     SELECT TABSCHEMA, TABNAME
     FROM SYSCAT.TABLES
     WHERE SECPOLICYID>0
     ```
   - For the Db2 Performance Optimization Feature:
     – Check whether there are any materialized query tables. Run the following command against every database in every instance in the Db2 copy:

     ```
     SELECT OWNER, TABNAME
     FROM SYSCAT.TABLES WHERE TYPE='S'
     ```
     – Check whether there are any multidimensional cluster tables. Run the following command against every database in every instance in the Db2 copy:

     ```
     SELECT A.TABSCHEMA, A.TABNAME, A.INDNAME, A.INDSCHEMA
     FROM SYSCAT.INDEXES A, SYSCAT.TABLES B
     WHERE (A.TABNAME=B.TABNAME AND A.TABSCHEMA=B.TABSCHEMA)
     AND A.INDEXTYPE='BLOK'
     ```
     – Check whether any of your instances use query parallelism (also known as *interquery parallelism*). Run the following command once in each instance in the Db2 copy:

     ```
     SELECT NAME, VALUE
     FROM SYSIBMADM.DBMCFG
     WHERE NAME IN ('intra_parallel')
     ```
     – Check if connection concentrator is enabled. Run the following command against every instance in the Db2 copy:

     ```
     db2 get dbm cfg
     ```

     This command displays the current values of database manager configuration parameters including MAX_CONNECTIONS and MAX_COORDAGENTS. If the value of the MAX_CONNECTIONS is greater than the value of the MAX_COORDAGENTS then connection concentrator is enabled. If you are not using Db2 Enterprise Server Edition, Db2 Advanced Enterprise Server Edition, or Db2 Connect Server products, ensure that connection concentrator is disabled. This is because connection concentrator is only supported for Db2 Enterprise Server Edition, Db2 Advanced Enterprise Server Edition, or Db2 Connect Server products.
   - For the Db2 Storage Optimization Feature:
     – Check if any tables have row level compression enabled. Run the following command against every database in every instance in the Db2 copy:

     ```
     SELECT TABSCHEMA, TABNAME
     FROM SYSCAT.TABLES
     WHERE COMPRESSION IN ('R', 'B')
     ```
     – Check if any indexes have compression enabled. Run the following command against every database in every instance in the Db2 copy:

```
SELECT TABSCHEMA, TABNAME, INDNAME, COMPRESSION
FROM SYSCAT.INDEXES
WHERE COMPRESSION = 'Y'
```

– Check if any compression dictionary still exists for a table that has row level compression deactivated. Run the following command against every database in every instance in the Db2 copy:

```
SELECT TABSCHEMA, TABNAME
FROM SYSIBMADM.ADMINTABINFO
WHERE DICTIONARY_SIZE <> 0 OR XML_DICTIONARY_SIZE <> 0
```

Note: This query might be resource intensive and might take a long time to run. Run this query only if Storage Optimization license violations are being reported even though there are no tables that have row level compression enabled.

Diagnosing and resolving locking problems

To resolve a locking problem, you need to start by diagnosing the type of lock event causing the SQL query performance slowdown, or query completion failure, and the SQL statement or statements involved. The steps to help in diagnosing the type of locking problem and the steps that can then be taken to help resolve the locking issue are provided here.

Introduction

A locking problem is the proper diagnosis if you are experiencing a failure of applications to complete their tasks or a slow down in the performance of SQL queries due to locks. Therefore, the ideal objective is not to have any lock timeouts or deadlocks on a database system, both of which result in applications failing to complete their tasks.

Lock waits are normal expected events, but if the time spent waiting for a lock becomes large, then lock waits can slow down both SQL query performance and completion of an application. Excessive lock wait durations have a risk of becoming lock timeouts which result in the application not completing its tasks.

Lock escalations are a consideration as a locking problem when they contribute to causing lock timeouts. Ideally, the objective is not to have any lock escalations, but a small number can be acceptable if adverse effects are not occurring.

It is suggested that you monitor lock wait, lock timeout, and deadlock locking events at all times; typically at the workload level for lock waits, and at the database level for lock timeouts and deadlocks.

The diagnosis of the type of locking problem that is occurring and its resolution begins with the collection of information and looking for diagnostic indicators. The following sections help to guide you through this process.

Collect information

In general, to be able to objectively assess that your system is demonstrating abnormal behavior which can include processing delays and poor performance, you must have information that describes the typical behavior (baseline) of your system. A comparison can then be made between your observations of suspected abnormal behavior and the baseline. Collecting baseline data, by scheduling periodic operational monitoring tasks, is a key component of the troubleshooting process.For more detailed information about establishing the baseline operation of your system, see:“ Operational monitoring of system performance” in *Troubleshooting and Tuning Database Performance*.

To confirm what type of locking problem is the reason for your SQL query performance slowdown or query completion failure, it is necessary to collect information that would help to identify what type of lock event is involved, which application is requesting or holding this lock, what was the application doing during this event, and the SQL statement or statements that are involved in being noticeably slow.

The creation of a locking event monitor, use of a table function, or use of the **db2pd** command can collect this type of information. The information gathered by the locking event monitor can be categorized into three main categories:
- Information about the lock in question
- Information about the application requesting this lock and its current activities. In the case of a deadlock, this is information about the statement referred to as the victim.
- Information about the application owning the lock and its current activities. In the case of a deadlock, this is information about the statement referred to as the participant.

For instructions about how to monitor lock wait, lock timeout, and deadlock locking events, see: “Monitoring locking events” in the *Database Monitoring Guide and Reference*.

Look for diagnostic indicators

The locking event monitor, a table function, or running the **db2pd** command can collect information that can help isolate the nature of a locking problem. Specifically, the following topics contain diagnostically indicative information to help you to diagnose and confirm the particular type of locking problem you are experiencing.
- If you are experiencing long wait times and no lock timeouts, then you likely have a lock wait problem. To confirm: “Diagnosing a lock wait problem.”
- If you are experiencing an increased number of deadlocks than the baseline number, then you likely have a deadlock problem. To confirm: “Diagnosing a deadlock problem” on page 157.
- If you are experiencing an increased number of lock timeouts and the **locktimeout** database configuration parameter is set to a nonzero time value, then you likely have a lock timeout problem. To confirm (also consider lock wait problem): “Diagnosing a lock timeout problem” on page 159
- If you are experiencing a higher than typical number of lock waits and the locking event monitor indicates that lock escalations are occurring (Yes), then you likely have a lock escalation problem. To confirm: “Diagnosing a lock escalation problem” on page 162

Related information:

Best practices: Troubleshooting Db2 servers

Diagnosing a lock wait problem:

A lock wait occurs when a transaction tries to obtain a lock on a resource that is already held by another transaction. When the duration of the lock wait time is extended, this results in a slow down of SQL query execution. You likely have a lock wait problem if you are experiencing long or unexpected lock wait times and no lock timeouts.

Before you begin

In general, to be able to objectively assess that your system is demonstrating abnormal behavior which can include processing delays and poor performance, you must have information that describes the typical behavior (baseline) of your system. A comparison can then be made between your observations of suspected abnormal behavior and the baseline. Collecting baseline data, by scheduling periodic operational monitoring tasks, is a key component of the troubleshooting process. For more detailed information about establishing the baseline operation of your system, see: ../com.ibm.db2.luw.admin.perf.doc/doc/c0054690.dita.

For instructions about how to monitor lock wait locking events, see: "Monitoring locking events" in *Database Monitoring Guide and Reference*.

About this task

Diagnosis

A lock wait occurs when one transaction (composed of one or more SQL statements) tries to acquire a lock whose mode conflicts with a lock held by another transaction. Excessive lock wait time often translates into poor response time, so it is important to monitor. The amount of lock wait time is best normalized to one thousand transactions because lock wait time on a single transaction is typically quite low and a normalized measurement is easier to handle.

There are different qualities of lock wait that must be taken into consideration when attempting to confirm diagnosis of each of them. The following list shows the three different qualities of lock wait and how best to diagnose them:

- Long individual lock wait
  - Check for peak lock wait time from service class and workload. Set up the locking event monitor on workload to obtain that value.
- Long lock wait time but short individual lock waits
  - Typically a result of a lock convoy. Use the **`db2pd -locks wait`** command to detect wait chains.
- Types of lock being waited on
  - Checking this might help determine the problem. Find the agent that is waiting on the lock to get information about the lock type. Use the lock type information to determine if something obvious is occurring. For example, a package lock might indicate a **`BIND`**/**`REBIND`** command or DDL colliding with a user of that package; an internal c (catalog cache) lock might indicate a DDL colliding with a statement compilation.

Indicative signs

Look for the following indicative signs of lock waits:

- The number of lock waits is increasing (increasing **`lock_waits`** monitor element value)

- A high percentage of active agents waiting on locks (for example, 20%, or more, of the total active agents). For information about how to obtain this information, see the next section, "What to monitor".
- An increasing value of lock wait time (**lock_wait_time** monitor element) captured at database or workload level

What to monitor

Unlike many other types of Db2 monitor data, locking information is very transient. Apart from **lock_wait_time**, which is a running total, most other lock information goes away when the locks themselves are released. Thus, lock and lock wait event data are most valuable if collected periodically over a period of time, so that the evolving picture can be better understood.

To collect information about active agents waiting on locks, use the WLM_GET_SERVICE_CLASS_AGENTS table function. Agents waiting for locks are indicated by agents with the following attribute-value pairs:

- EVENT_OBJECT = LOCK
- EVENT_TYPE = ACQUIRE

You can also use application snapshot, the lock administrative views, or the lock wait option of the **db2pd -wlocks** command to obtain information about active agents waiting on locks.

These are the key indicator monitoring elements:

- **lock_waits** value is increasing
- long **lock_wait_time** value

If you have observed one or more of the indicative signs listed here, then you are likely experiencing a problem with lock waits. Follow the link in the "What to do next" section to resolve this issue.

What to do next

After having diagnosed that lock waits are likely causing the problem you are experiencing, take steps to resolve the issue: "Resolving lock wait problems"

Resolving lock wait problems:

After diagnosing a lock wait problem, the next step is to attempt to resolve the issue resulting from an application having to wait too long for a lock. Guidelines are provided here to help you resolve lock wait problems and assist you in preventing such incidents from occurring in future.

Before you begin

Confirm that you are experiencing a lock wait problem by taking the necessary diagnostic steps for locking problems outlined in "Diagnosing and resolving locking problems" on page 152.

About this task

The guidelines provided here can help you to resolve the lock wait problem you are experiencing and help you to prevent such future incidents.

Procedure

Use the following steps to diagnose the cause of the unacceptable lock wait problem and to apply a remedy:

1. Obtain information from the administration notification log about all tables where agents are spending long periods of time waiting for locks.
2. Use the information in the administration notification log to decide how to resolve the lock wait problem. There are a number of guidelines that help to reduce lock contention and lock wait time. Consider the following options:
   - If possible, avoid very long transactions and WITH HOLD cursors. The longer locks are held, the more chance that they cause contention with other applications. This is only an issue if you are using a high isolation level.
   - It is best practice to commit the following actions as soon as possible:
     - Write actions such as delete, insert, and update
     - Data definition language (DDL) statements, for example ALTER, CREATE, and DROP statements
     - **BIND** and **REBIND** commands
   - After issuing ALTER or DROP DDL statements, run the SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS procedure to revalidate any data objects and the **db2rbind** command to rebind any packages.
   - Avoid fetching result sets that are larger than necessary, especially under the repeatable read (RR) isolation level. The more that rows are touched, the more locks are held, and the greater the opportunity to run into a lock that is held by someone else. In practical terms, this often means pushing down row selection criteria into a WHERE clause of the SELECT statement, rather than bringing back more rows and filtering them at the application. For example:

     ```
     exec sql declare curs for
       select c1,c2 from t
       where c1 not null;
     exec sql open curs;
     do {
       exec sql fetch curs
         into :c1, :c2;
     } while( P(c1) != someVar );

     ==>

     exec sql declare curs for
       select c1,c2 from t
       where c1 not null
       and myUdfP(c1) = :someVar;
     exec sql open curs;
     exec sql fetch curs
         into :c1, :c2;
     ```
   - Avoid using higher isolation levels than necessary. Repeatable read might be necessary to preserve result set integrity in your application; however, it does incur extra cost in terms of locks held and potential lock conflicts.
   - If appropriate for the business logic in the application, consider modifying locking behavior through the **DB2_EVALUNCOMMITTED**, **DB2_SKIPDELETED**, and **DB2_SKIPINSERTED** registry variables. These registry variables enable Db2 database manager to delay or avoid taking locks in some circumstances, thereby reducing contention and potentially improving throughput.
   - Eliminate lock escalations wherever possible.

What to do next

Rerun the application or applications to ensure that the locking problem has been eliminated by checking the administration notification log for lock-related entries or checking the lock wait and lock wait time metrics for the appropriate workload, connection, service subclass, unit of work, and activity levels.

Diagnosing a deadlock problem:

A deadlock is created when two applications lock data that is needed by the other, resulting in a situation in which neither application can continue executing without the intervention of the deadlock detector. The deadlock slows down the participant transaction while it waits for deadlock detection, wastes system resources by rolling back the victim transaction, and causes extra system work and transaction log access during the whole process. You likely have a deadlock problem if you are experiencing an increased number of deadlocks than the baseline number and transactions are being re-executed.

Before you begin

In general, any observed deadlock is considered abnormal. To be able to objectively assess that your system is demonstrating abnormal behavior which can include processing delays and poor performance, you must have information that describes the typical behavior (baseline) of your system. A comparison can then be made between your observations of suspected abnormal behavior and the baseline. Collecting baseline data, by scheduling periodic operational monitoring tasks, is a key component of the troubleshooting process. .

For instructions about how to monitor deadlock locking events, see: "Monitoring locking events" in *Database Monitoring Guide and Reference*.

About this task

Diagnosis

A deadlock is created when two applications lock data that is needed by the other, resulting in a situation in which neither application can continue executing without the intervention of the deadlock detector. The victim application has to re-execute the transaction from the beginning after the system automatically rolls back the previous deadlocked transaction. Monitoring the rate at which this happens helps avoid the case where many deadlocks drive significant extra load on the system without the DBA being aware.

Indicative signs

Look for the following indicative signs of deadlocks:

- One or more applications are occasionally re-executing transactions
- Deadlock message entries in the administration notification log
- Increased number of deadlocks displayed for the **`deadlocks`** monitor element
- Increased number of roll backs displayed for the **`int_deadlock_rollbacks`** monitor element
- Increased amount of time an agent spends waiting for log records to be flushed to disk which is displayed for the **`log_disk_wait_time`** monitor element

What to monitor

The cost of a deadlock varies, and is directly proportional to the length of the rolled-back transaction. All the same, any deadlock generally indicates a problem.

There are essentially three approaches to detecting deadlock events:

1. Set a locking event monitor and set the **`mon_deadlock`** database configuration parameter to capture details on all deadlock events that occur database-wide
2. Monitor the administration notification log for deadlock messages and basic information that accompanies them

   Note: To enable deadlock messages to be written to the administration notification log file, set the **`mon_lck_msg_lvl`** database configuration parameter to a value of 2.
3. Monitor the indicator monitoring elements by way of a table function

Most users adopt the first approach. By monitoring the key indicator monitor elements to detect when a deadlock occurs, users can then obtain detailed information by checking the information collected by the event monitor.

These are the key indicator monitoring elements:

- **`deadlocks`** value is non-zero
- **`int_deadlock_rollbacks`** shows increase in number of roll backs due to deadlock events
- **`log_disk_wait_time`** shows increase in amount of time agents spend waiting for logs to be flushed to disk

If you have observed one or more of the indicative signs listed here, then you are likely experiencing a problem with deadlocks. Follow the link in the "What to do next" section to resolve this issue.

What to do next

After having diagnosed that deadlocks are likely causing the problem you are experiencing, take steps to resolve the issue: "Resolving deadlock problems"

Resolving deadlock problems:

After diagnosing a deadlock problem, the next step is to attempt to resolve the deadlock issue resulting between two concurrently running applications each of which have locked a resource the other application needs. The guidelines provided here can help you to resolve the deadlock problem you are experiencing and help you to prevent such future incidents.

Before you begin

Confirm that you are experiencing a deadlock problem by taking the necessary diagnostic steps for locking problems outlined in "Diagnosing and resolving locking problems" on page 152.

About this task

The guidelines provided here can help you to resolve the deadlock problem you are experiencing and help you to prevent such future incidents.

Procedure

Use the following steps to diagnose the cause of the unacceptable deadlock problem and to apply a remedy:

1. Obtain information from the lock event monitor or administration notification log about all tables where agents are experiencing deadlocks.
2. Use the information in the administration notification log to decide how to resolve the deadlock problem. There are a number of guidelines that help to reduce lock contention and lock wait time. Consider the following options:
   - Each application connection should process its own set of rows to avoid lock waits.
   - Deadlock frequency can sometimes be reduced by ensuring that all applications access their common data in the same order - meaning, for example, that they access (and therefore lock) rows in Table A, followed by Table B, followed by Table C, and so on. If two applications take incompatible locks on the same objects in different order, they run a much larger risk of deadlocking.
   - A lock timeout is not much better than a deadlock, because both cause a transaction to be rolled back, but if you must minimize the number of deadlocks, you can do it by ensuring that a lock timeout will usually occur before a potential related deadlock can be detected. To do this, set the value of the **locktimeout** database configuration parameter (units of seconds) to be much lower than the value of the **dlchktime** database configuration parameter (units of milliseconds). Otherwise, if **locktimeout** is longer than the **dlchktime** interval, the deadlock detector could wake up just after the deadlock situation began, and detect the deadlock before the lock timeout occurs.
   - Avoid concurrent DDL operations if possible. For example, DROP TABLE statements can result in a large number of catalog updates as rows might have to be deleted for the table indexes, primary keys, check constraints, and so on, in addition to the table itself. If other DDL operations are dropping or creating objects, there can be lock conflicts and even occasional deadlocks.
   - It is best practice to commit the following actions as soon as possible:
     - Write actions such as delete, insert, and update
     - Data definition language (DDL) statements, such as ALTER, CREATE, and DROP
     - **BIND** and **REBIND** commands
3. The deadlock detector is unable to know about and resolve the following situation, so your application design must guard against this. An application, particularly a multithreaded one, can have a deadlock involving a Db2 lock wait and a wait for a resource outside of the Db2 software, such as a semaphore. For example, connection A can be waiting for a lock held by connection B, and B can be waiting for a semaphore held by A.

What to do next

Rerun the application or applications to ensure that the locking problem has been eliminated by checking the administration notification log for lock-related entries.

Diagnosing a lock timeout problem:

A lock timeout occurs when a transaction, waiting for a resource lock, waits long enough to have surpassed the wait time value specified by the **locktimeout** database configuration parameter. This consumes time which causes a slow down in SQL query performance. You likely have a lock timeout problem if you are experiencing an increased number of lock timeouts and the **`locktimeout`** database configuration parameter is set to a nonzero time value.

Before you begin

In general, to be able to objectively assess that your system is demonstrating abnormal behavior which can include processing delays and poor performance, you must have information that describes the typical behavior (baseline) of your system. A comparison can then be made between your observations of suspected abnormal behavior and the baseline. Collecting baseline data, by scheduling periodic operational monitoring tasks, is a key component of the troubleshooting process. .

For instructions about how to monitor lock timeout locking events, see: "Monitoring locking events" in *Database Monitoring Guide and Reference*.

About this task

Diagnosis

Sometimes, lock wait situations lead to lock timeouts that cause transactions to be rolled back. The period of time until a lock wait leads to a lock timeout is specified by the database configuration parameter **`locktimeout`**. Lock timeouts, in excessive numbers, can be as disruptive to a system as deadlocks. Although deadlocks are comparatively rare in most production systems, lock timeouts can be more common. The application usually has to handle them in a similar way: re-executing the transaction from the beginning. Monitoring the rate at which this happens helps avoid the case where many lock timeouts drive significant extra load on the system without the DBA being aware.

Indicative signs

Look for the following indicative signs of lock timeouts:

- An application is frequently re-executing transactions
- **`lock_timeouts`** monitor element value is climbing
- Lock timeout message entries in the administration notification log

What to monitor

Due to the relatively transient nature of locking events, lock event data is most valuable if collected periodically over a period of time, so that the evolving picture can be better understood.

You can monitor the administration notification log for lock timeout messages.

Note: To enable lock timeout messages to be written to the administration notification log file, set the **`mon_lck_msg_lvl`** database configuration parameter to a value of 3.

Create an event monitor to capture lock timeout data for a workload or database.

These are the key indicator monitoring elements:

- **`lock_timeouts`** value is climbing
- **`int_rollbacks`** value is climbing

If you have observed one or more of the indicative signs listed here, then you are likely experiencing a problem with lock timeouts. Follow the link in the "What to do next" section to resolve this issue.

What to do next

After having diagnosed that lock timeouts are likely causing the problem you are experiencing, take steps to resolve the issue: "Resolving lock timeout problems"

Resolving lock timeout problems:

After diagnosing a lock timeout problem, the next step is to attempt to resolve the issue resulting from an application or applications waiting for locks until the lock timeout period has elapsed. The guidelines provided here can help you to resolve the lock timeout problem you are experiencing and help you to prevent such future incidents.

Before you begin

Confirm that you are experiencing a lock timeout problem by taking the necessary diagnostic steps for locking problems outlined in "Diagnosing and resolving locking problems" on page 152.

About this task

The guidelines provided here can help you to resolve the lock timeout problem you are experiencing and help you to prevent such future incidents.

Procedure

Use the following steps to diagnose the cause of the unacceptable lock timeout problem and to apply a remedy:

1. Obtain information from the lock event monitor or administration notification log about all tables where agents are experiencing lock timeouts.
2. Use the information in the administration notification log to decide how to resolve the lock timeout problem. There are a number of guidelines that help to reduce lock contention and lock wait time that can result in a reduced number of lock timeouts. Consider the following options:
   - Tune the **`locktimeout`** database configuration parameter to a number of seconds appropriate for your database environment.
   - If possible, avoid very long transactions and WITH HOLD cursors. The longer locks are held, the more chance that they cause contention with other applications.
   - It is best practice to commit the following actions as soon as possible:
     - Write actions such as delete, insert, and update
     - Data definition language (DDL) statements, such as ALTER, CREATE, and DROP
     - **`BIND`** and **`REBIND`** commands
   - Avoid fetching result sets that are larger than necessary, especially under the repeatable read (RR) isolation level. The more that rows are touched, the more locks are held, and the greater the opportunity to run into a lock that is held by someone else. In practical terms, this often means pushing down row

selection criteria into a WHERE clause of the SELECT statement, rather than bringing back more rows and filtering them at the application. For example:

```
exec sql declare curs for
  select c1,c2 from t
  where c1 not null;
exec sql open curs;
do {
  exec sql fetch curs
    into :c1, :c2;
} while( P(c1) != someVar );

==>

exec sql declare curs for
  select c1,c2 from t
  where c1 not null
  and myUdfP(c1) = :someVar;
exec sql open curs;
exec sql fetch curs
    into :c1, :c2;
```

- Avoid using higher isolation levels than necessary. Repeatable read might be necessary to preserve result set integrity in your application; however, it does incur extra cost in terms of locks held and potential lock conflicts.
- If appropriate for the business logic in the application, consider modifying locking behavior through the **DB2_EVALUNCOMMITTED**, **DB2_SKIPDELETED**, and **DB2_SKIPINSERTED** registry variables. These registry variables enable Db2 database manager to delay or avoid taking locks in some circumstances, thereby reducing contention and potentially improving throughput.

What to do next

Rerun the application or applications to ensure that the locking problem has been eliminated by checking the administration notification log for lock-related entries or checking the lock wait and lock wait time metrics for the appropriate workload, connection, service subclass, unit of work, and activity levels.

Diagnosing a lock escalation problem:

A lock escalation occurs when, in the interest of reducing memory that is allocated to locks (lock space), numerous row-level locks are escalated into a single, memory-saving table lock. This situation, although automated and saves memory space devoted to locks, can reduce concurrency to an unacceptable level. You likely have a lock escalation problem if you are experiencing a higher than typical number of lock waits and the administration notification log entries indicate that lock escalations are occurring.

Before you begin

In general, to be able to objectively assess that your system is demonstrating abnormal behavior which can include processing delays and poor performance, you must have information that describes the typical behavior (baseline) of your system. A comparison can then be made between your observations of suspected abnormal behavior and the baseline. Collecting baseline data, by scheduling periodic operational monitoring tasks, is a key component of the troubleshooting process. For more detailed information about establishing the baseline operation of your system, see: ../com.ibm.db2.luw.admin.perf.doc/doc/c0054690.dita.

About this task

Diagnosis

Lock escalation from multiple row-level locks to a single table-level lock can occur for the following reasons:

- The total amount of memory consumed by many row-level locks held against a table exceeds the percentage of total memory allocated for storing locks
- The lock list runs out of space. The application that caused the lock list to be exhausted will have its locks forced through the lock escalation process, even though the application is not the holder of the most locks.

The threshold percentage of total memory allocated for storing locks, that has to be exceeded by an application for a lock escalation to occur, is defined by the **`maxlocks`** database configuration parameter and the allocated memory for locks is defined by the **`locklist`** database configuration parameter. In a well-configured database, lock escalation is rare. If lock escalation reduces concurrency to an unacceptable level, you can analyze the problem and decide on the best course of action.

Lock escalation is less of an issue, from the memory space perspective, if self tuning memory manager (STMM) is managing the memory for locks that is otherwise only allocated by the **`locklist`** database configuration parameter. STMM will automatically adjust the memory space for locks if it ever runs out of free memory space.

Indicative signs

Look for the following indicative signs of lock escalations:

- Lock escalation message entries in the administration notification log

What to monitor

Due to the relatively transient nature of locking events, lock event data is most valuable if collected periodically over a period of time, so that the evolving picture can be better understood.

Check this monitoring element for indications that lock escalations might be a contributing factor in the SQL query performance slow down:

- **`lock_escals`**

If you have observed one or more of the indicative signs listed here, then you are likely experiencing a problem with lock escalations. Follow the link in the "What to do next" section to resolve this issue.

What to do next

After having diagnosed that lock escalations are likely causing the problem you are experiencing, take steps to resolve the issue: "Resolving lock escalation problems"

Resolving lock escalation problems:

After diagnosing a lock escalation problem, the next step is to attempt to resolve the issue resulting from the database manager automatically escalating locks from row level to table level. The guidelines provided here can help you to resolve the lock escalation problem you are experiencing and help you to prevent such future incidents.

Before you begin

Confirm that you are experiencing a lock escalation problem by taking the necessary diagnostic steps for locking problems outlined in "Diagnosing and resolving locking problems" on page 152.

About this task

The guidelines provided here can help you to resolve the lock escalation problem you are experiencing and help you to prevent such future incidents.

The objective is to minimize lock escalations, or eliminate them, if possible. A combination of good application design and database configuration for lock handling can minimize or eliminate lock escalations. Lock escalations can lead to reduced concurrency and potential lock timeouts, so addressing lock escalations is an important task. The **lock_escals** monitor element and messages written to the administration notification log can be used to identify and correct lock escalations.

First, ensure that lock escalation information is being recorded. Set the value of the **mon_lck_msg_lvl** database configuration parameter to 1. This is the default setting. When a lock escalation event occurs, information regarding the lock, workload, application, table, and error SQLCODEs are recorded. The query is also logged if it is a currently executing dynamic SQL statement.

Procedure

Use the following steps to diagnose the cause of the unacceptable lock escalation problem and to apply a remedy:

1. Gather information from the administration notification log about all tables whose locks have been escalated and the applications involved. This log file includes the following information:
   - The number of locks currently held
   - The number of locks needed before lock escalation is completed
   - The table identifier and table name of each table being escalated
   - The number of non-table locks currently held
   - The new table-level lock to be acquired as part of the escalation. Usually, an S or X lock is acquired.
   - The internal return code that is associated with the acquisition of the new table-level lock
2. Use the administration notification log information about the applications involved in the lock escalations to decide how to resolve the escalation problems. Consider the following options:
   - You can enable the **DB2_AVOID_LOCK_ESCALATION** registry variable to return SQL0912N to the application, instead of performing lock escalation. The application then has an opportunity to either **COMMIT** or **ROLLBACK** which will release the locks held by this application.
   - Check and possibly adjust either the **maxlocks** or **locklist** database configuration parameters, or both. In a partitioned database system, make this change on all database partitions. The value of the **locklist** configuration parameter may be too small for your current workload. If multiple applications are experiencing lock escalation, this could be an indication that the lock list size needs to be increased. Growth in workloads or the addition of new applications could cause the lock list to be too small.

If only one application is experiencing lock escalations, then adjusting the **maxlocks** configuration parameter could resolve this. However, you may want to consider increasing **locklist** at the same time you increase **maxlocks** - if one application is allowed to use more of the lock list, all the other applications could now exhaust the remaining locks available in the lock list and experience escalations.

- You might want to consider the isolation level at which the application and the SQL statements are being run, for example RR, RS, CS, or UR. RR and RS isolation levels tend to cause more escalations because locks are held until a COMMIT is issued. CS and UR isolation levels do not hold locks until a COMMIT is issued, and therefore lock escalations are less likely. Use the lowest possible isolation level that can be tolerated by the application.
- Increase the frequency of commits in the application, if business needs and the design of the application allow this. Increasing the frequency of commits reduces the number of locks that are held at any given time. This helps to prevent the application from reaching the **maxlocks** value, which triggers a lock escalation, and helps to prevent all the applications from exhausting the lock list.
- You can modify the application to acquire table locks using the LOCK TABLE statement. This is a good strategy for tables where concurrent access by many applications and users is not critical; for example, when the application uses a permanent work table (for example, not a DGTT) that is uniquely named for this instance of the application. Acquiring table locks would be a good strategy in this case as it will reduce the number of locks being held by the application and increase the performance because row locks no longer need to be acquired and released on the rows that are accessed in the work table.

  If the application does not have work tables and you cannot increase the values for **locklist** or **maxlocks** configuration parameters, then you can have the application acquire a table lock. However, care must be taken in choosing the table or tables to lock. Avoid tables that are accessed by many applications and users because locking these tables will lead to concurrency problems which can affect response time, and, in the worst case, can lead to applications experiencing lock timeouts.

What to do next

Rerun the application or applications to ensure that the locking problem has been eliminated by checking the administration notification log for lock-related entries.

Troubleshooting SQL performance

Individual SQL queries run well, but performance decreases when running multiple queries:

There are a number of steps that can be taken to improve the performance of SQL queries that run slowly as a group.

A query might perform well when run by itself in a test environment, but then run slowly in a production environment with other queries running at the same time.

Symptoms
An SQL query runs reasonably well on its own, but then slows down in a production system when other queries are running at the same time. This may occur every day at a specific time, or seemingly at random.

Looking at the SQL statement there are no apparent reasons for the intermittent performance problem.

Causes

Often these types of performance issues are a result of sort heap memory allocation problems.

Sort heap memory is used by a range of queries (the list is included in the **`sortheap`** documentation), so it is important to consider the settings of the following configuration parameters:

- Sort heap size (**`sortheap`**) which specifies the amount of memory allocated for each individual sort
- Sort heap threshold (**`sheapthres`**) which specifies the total amount of memory that is available for all the sorts running across the instance

As the number of queries running at the same time increases, the system might reach the **`sheapthres`** threshold, resulting in performance issues.

Resolving the problem

In many cases, the Db2 self-tuning memory manager (STMM) can adjust the sort memory allocation to keep queries running efficiently. See the **`self_tuning_mem`** database configuration parameter documentation for more details. If the system is not running STMM then consider the following options:

Modifying SQL

It may be possible to reduce the amount of sort heap memory used by modifying the SQL:

1. Are all the appropriate indexes created for the query? If a column is known to contain unique values, for example, creating a unique index may reduce the need for sort heap memory.
2. Determine whether one or more of the SQL statements can be modified to use less sort memory. Is each of the ORDER BY clauses required in the query? Can GROUP BY be used instead of DISTINCT in a subquery?
3. Make use of column group statistics or statistical views, which in some cases enable the Db2 query optimizer to determine an access plan that uses less sort heap memory.

Increasing `sheapthres` size

If none of the previous changes to the SQL are possible, or do not improve performance, then increasing the value of the **`sheapthres`** parameter is the next consideration. This parameter defines the total amount of memory used by all the private sorts running at any time. When the total private sort memory consumption for an instance reaches this limit, the memory allocated for additional incoming private sort requests is considerably reduced. The **`post_threshold_sorts`** monitor element is used to track sort requests that must run with reduced memory once the **`sheapthres`** threshold has been exceeded.

See the **`sheapthres`** documentation for details on how to increase the amount of memory allocated for all sorts, and how to balance SQL sort performance with memory usage. You will need to consider the amount of memory set aside for any one sort (**`sortheap`**) and the typical number of sorts that might occur on the system when performance begins to drop. To verify the effectiveness of increasing the value of **`sheapthres`** you should see a decrease in the number of **`post_threshold_sorts`** after the change.

Decreasing `sortheap` size

Finally, in some situations where you are unable to increase the overall sort heap memory (**`sheapthres`**) and you have many small sorts, it may be beneficial to decrease the individual sort heap size **`sortheap`** parameter instead. This results in less memory allocated for each individual sort, but an increase in the number of sorts that can run at any one time.

You should only consider this option, however, if you can verify that you have many small sorts, each small enough to run within a smaller sort heap. If you set **`sortheap`** too small then the sorts will start spilling to disk, which slows down performance because disk I/O is much slower than in-memory operations. You can use the `NumSpills` value returned as part of the **`-sort`** parameter when running the **`db2pd`** command to determine whether the individual sort heap size **`sortheap`** is set too small for the queries.

Troubleshooting optimization

Troubleshooting optimization guidelines and profiles:

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

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

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

The DDLs used to create the diagnostic explain tables is shown in Figure 5 on page 168.

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

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

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

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

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

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

Figure 5. DDLs used to create the diagnostic explain tables

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

Troubleshooting partitioned database environments

FCM problems related to 127.0.0.2 (Linux and UNIX):

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

Symptoms

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

Causes

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

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

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

Environment

The problem is limited to partitioned database environments.

Resolving the problem

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

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

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

The AIX operating system supports the ability to encrypt a JFS2 file system or set of files. This feature is not supported with partitioned database environments in Db2 database products. An SQL10004C error will occur if you attempt to create a partitioned database environment using EFS (encrypted file systems) on AIX.

Symptoms

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

Causes

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

Diagnosing the problem

The Db2 diagnostic (**`db2diag`**) log files will contain the error message and the following text: `OSERR : ENOATTR (112) "No attribute found"`.

Resolving the problem

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

Troubleshooting table states during data redistribution:

Before starting a redistribution operation, ensure that all tables in the database partition group are in full access mode and are in normal state.

Symptoms
If table states cause the redistribution to fail, the error message indicates that the database partition group cannot be redistributed or that the operation is not allowed. For example, SQL02436N, SQL6056N, and SQL0668N messages can be symptoms of this problem.

Note: If the error message lists a table name, it might not be the only problematic table in the database partition group. By troubleshooting the table states for all of the tables in the database partition group, you can avoid multiple unsuccessful redistribution attempts.

Diagnosing the problem
User response:

1. Determine which tables are in an inoperative state (SYSCAT.TABLES.STATUS='X').

   Issue the following query:

   ```
   SELECT TABNAME
      FROM SYSCAT.TABLES AS TABLES, SYSCAT.TABLESPACES AS TABLESPACES
      WHERE TABLES.TBSPACE = TABLESPACES.TBSPACE AND TABLES.STATUS = 'X'
      AND TABLESPACES.DBPGNAME = 'IBMDEFAULTGROUP'
   ```

   where *IBMDEFAULTGROUP* is the database partition group name.
2. Determine which tables are in the "set integrity pending" state (SYSCAT.TABLES.STATUS='C').

   Issue the following query:

   ```
   SELECT TABNAME
      FROM SYSCAT.TABLES AS TABLES, SYSCAT.TABLESPACES AS TABLESPACES
      WHERE TABLES.TBSPACE = TABLESPACES.TBSPACE AND TABLES.STATUS = 'C'
      AND TABLESPACES.DBPGNAME = 'IBMDEFAULTGROUP'
   ```

   where *IBMDEFAULTGROUP* is the database partition group name.
3. Determine which tables are in a normal state, but are not in full access mode.

   Issue the following query:

   ```
   SELECT DISTINCT TRIM(T.OWNER) || \'.\' || TRIM(T.TABNAME)
   AS NAME, T.ACCESS_MODE, A.LOAD_STATUS
   FROM SYSCAT.TABLES T, SYSCAT.DBPARTITIONGROUPS N,
   SYSIBMADM.ADMINTABINFO A
   WHERE T.PMAP_ID = N.PMAP_ID
   AND A.TABSCHEMA = T.OWNER
   AND A.TABNAME = T.TABNAME
   AND N.DBPGNAME = 'IBMDEFAULTGROUP'
   AND (T.ACCESS_MODE <> 'F' OR A.LOAD_STATUS IS NOT NULL)
   AND T.STATUS='N'
   ```

where *IBMDEFAULTGROUP* is the database partition group name. If this query takes a long time to execute, terminate the query, issue the **RUNSTATS** command on all of the tables involved in this query, and then reissue the query.

Resolving the problem
User response:

1. Take the required corrective action:
   a. For each table in an inoperative state, use the **LOAD QUERY** command to determine the specific table state.
      - For tables in "load in progress" state, wait until the load operation has finished. You can use the **LOAD QUERY** command to monitor the progress the load operation.
      - For tables in "load pending" state, restart or terminate the previously failed load operation by issuing the **LOAD** command with **RESTART** or **TERMINATE** command parameters.
      - For tables in "read access only" state, use the **LOAD QUERY** command to check whether the table is in the process of being loaded. If yes, wait until the load utility has completed, or if necessary, restart or terminate the previously failed load operation. If a load operation is currently not in progress, issue the SET INTEGRITY statement with the IMMEDIATE CHECKED option. This action validates the constraints in the newly loaded portion of the table.
      - For tables in "reorg pending" state, perform a classic reorg operation to make the table accessible.
      - For tables in "not load restartable" state, issue a **LOAD TERMINATE** or **LOAD REPLACE** command.
      - For tables in "unavailable" state, drop the table or restore it from a backup.

   b. For the tables in "set integrity pending" state, execute the SET INTEGRITY statement with the IMMEDIATE CHECKED option.
   c. For the tables that are not in full access mode, execute the SET INTEGRITY statement with the IMMEDIATE CHECKED option on the dependent immediate materialized query and staging tables.

   Note: You can also choose to defer resolving the problem and temporarily omit the tables from the redistribution operation by specifying the **EXCLUDE** parameter in the **REDISTRIBUTE DATABASE PARTITION GROUP** command. In this case, the redistribution operation completes successfully, but the database partition group is only partially redistributed. As a result, redistributed tables in the database partition group could use a different partitioning map than tables that have not been redistributed. Also, if collocation between redistributed and non-redistributed tables existed before the redistribute operation, the collocation property between these tables is temporarily disabled. Query performance might not be optimal until a full redistribution occurs.
2. If a previous redistribution operation failed, resubmit it using either the **CONTINUE** or **ABORT** option. **CONTINUE** completes the previously aborted redistribution operation and **ABORT** undoes the effects of the previously aborted operation.

Troubleshooting scripts

Troubleshooting scripts:

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

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

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

Troubleshooting the Db2 pureScale Feature

This troubleshooting section contains information that will help you to understand, isolate, and resolve problems that are specific to the IBM Db2 pureScale Feature. The information found here complements the Db2 troubleshooting and the Db2 pureScale monitoring information found in the Db2 Information Center.

Note: If you require support for any IBM product that is packaged as part of the Db2 pureScale software stack, then open a service request or problem management record (PMR) for the Db2 pureScale Feature. Opening a PMR for the Db2 pureScale Feature helps resolve problems more efficiently.

How to diagnose a problem with the Db2 pureScale Feature

When you encounter a problem, it is important to define and isolate the problem accurately before you attempt to resolve it. Use the diagnostic steps in this topic to guide yourself through the problem-determination process with the troubleshooting tools that are included with the Db2 pureScale Feature. In many cases, you might be able to resolve the problem yourself.

The major areas covered in the troubleshooting documentation include the following ones:

- Obtaining diagnostic information, both logs and traces
- Installation, getting started with, and uninstallation of the Db2 pureScale Feature
- Component failures, such as host or members failures, including how to identify and resolve states, alerts, and restart conditions
- Communication failures (uDAPL)
- Cluster file system problems and failures (GPFS™)
- Cluster manager software failures (Tivoli SA MP)
- Problem scenarios where you should call IBM Service

Is it a current problem or a recurring problem?

Once you have installed the Db2 pureScale Feature and are up and running, problems broadly fall into one of two categories: Problems that affect your server right now, and problems that occurred at some point in the past but were probably recovered from automatically by the Db2 pureScale Feature.

Problems that affect your server right now
These problems typically exhibit one or several readily apparent symptoms that affect your server in some way. For example, you might receive user reports that there is a performance slowdown or you might be able to observe that there is a loss of capacity in the system yourself. Overall, the system continues to be available. Such symptoms are indicative of a problem with a member or a cluster caching facility (CF), which you can investigate directly.

If you observe a complete system outage, there are several possible culprits, but the first one you should check is whether there is at least one CF up and running to provide essential infrastructure services to the instance. Without a CF, there is no component available to handle locking and caching for members, and the instance cannot run.

Problems that occurred at some point in the past
These problems typically do not appear to affect your server now, but you might have some indication that there was a problem in the past. For example, you might see some unexplained log entries that point to something going on in your system that should be diagnosed. The highly available and fault tolerant character of the Db2 pureScale Feature can mask some problems, because the instance continues to recover from most component failures without requiring your intervention, even if a component fails repeatedly. Effectively, you need to monitor for intermittent or recurring problems over time to determine whether or not there exists a hidden problem on your server that keeps getting fixed by itself only to recur later on, and then resolve the underlying cause.

Diagnostic information you should look at

You need to understand when and why the problem occurs, based on your problem definition and the diagnostic data available to you. As you go through the diagnostic data, look for events or alerts that correlate the symptoms in your problem definition with a possible cause.

The Db2 pureScale-specific commands you use for troubleshooting can be run from any member or cluster caching facility host to obtain instance-wide information. Even if the database is unavailable or the instance is stopped, reporting of some data is often still possible, as long as the Db2 pureScale cluster manager component is available on the host where you issue the command.

The recommended sequence of diagnostic steps for the Db2 pureScale Feature is:
1. Issue the **db2instance -list** command to identify current issues affecting the instance. Check for alerts that affect host computers first, then check the CFs and the members for alerts. You can find the current state of members, cluster caching facilities, and hosts in the output, which will indicate whether alerts, errors or recovery-related states exist.
2. Issue the **db2cluster -cm -list -alert** command to check for recommended actions for these alerts. The output of the **db2cluster** command will tell you what alerts exist, what the effects of the alert conditions are, and what you need to do to resolve them. See if you can resolve the problem through the suggested action. You can also clear alerts with this command.

3. Check the diagnostic logs available to you for any hints that might indicate a possible cause for the problem. If you know roughly when the problem started, you can narrow down your search by checking log entries with a corresponding timestamp.
   - Check the `DIAGPATH` file path for any diagnostic files produced and check the FODC directories
   - Check the `CF_DIAGPATH` file path for any cluster caching facility diagnostic data produced
   - Check the `db2diag.log` diagnostic log file for recent diagnostic information and error messages
   - Check the notification log for recent messages than might indicate the starting point of the problem
   - Run the **`errpt -a`** command to view the system error log. Look for log entries that correlate with the time of failure

Next steps

The result of stepping through the available diagnostic information will determine what troubleshooting scenarios you should look at next. Once you have narrowed down the scope of the problem, you can navigate through the Db2 pureScale troubleshooting documentation to find the context that most likely applies to your problem definition. Often, you will find two types of troubleshooting content, very context-specific answers to frequently asked troubleshooting questions (FAQs), and more comprehensive troubleshooting scenarios that show you how to interpret the diagnostic data and resolve the problem

For example, if the diagnostic information you looked at shows that a member has an alert and is waiting to fail back to its home host after having failed over to a guest host for recovery purposes (an operation known as a restart light), you can locate the associated failure scenario in the troubleshooting documentation by looking at the Db2 pureScale instance operation scenarios, and then at the subsection for a members or host with alert. Not all possible failure scenarios are covered, but many are.

Understanding the Db2 pureScale Feature resource model

The new Db2 pureScale Feature resource model represents cluster caching facilities and the shared cluster file system.

The Db2 pureScale Feature resource model differs from the resource model used in previous Db2 Versions.

The new resource model implemented with Db2 pureScale Feature is necessary to represent cluster caching facilities (CFs) and the shared clustered file system. In a Db2 pureScale shared data instance, one CF fulfills the primary role, which contains the currently active data for the shared data instance. The second CF maintains a copy of pertinent information for immediate recovery of the primary role.

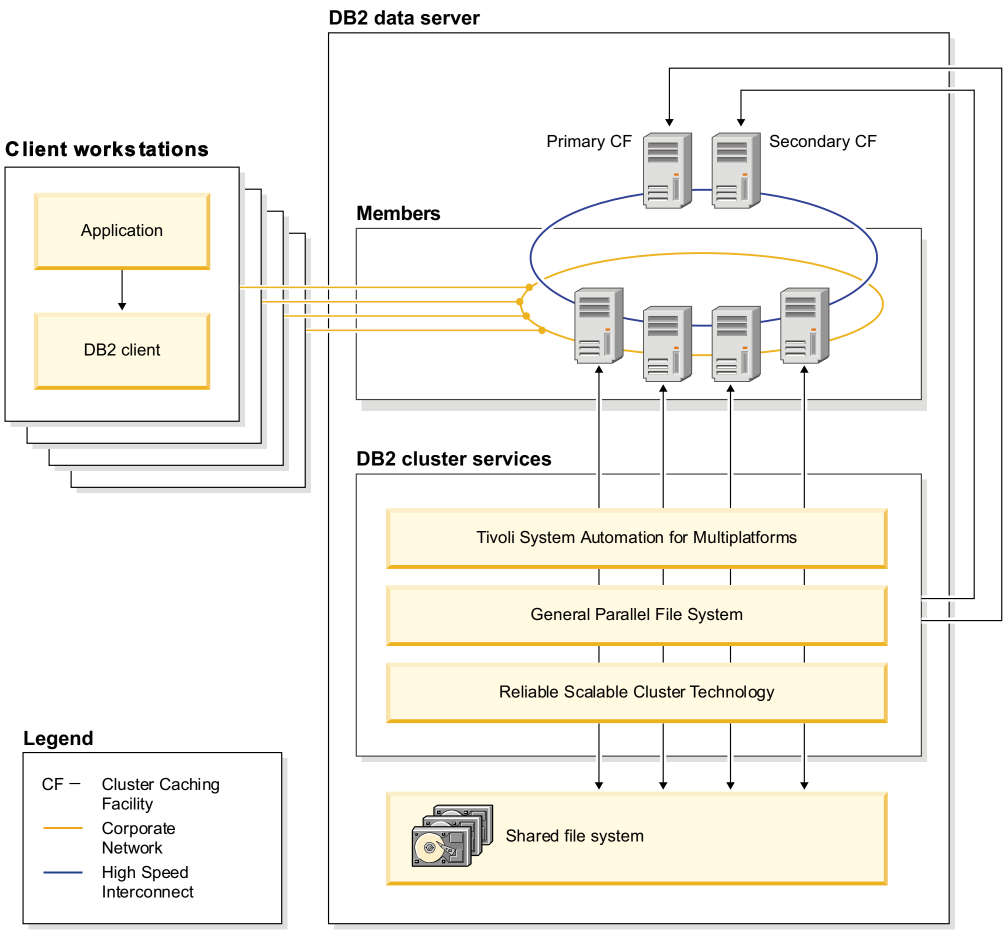
The new resource model allows IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) to appropriately automate the movement of the primary role in case of failure of the primary CF node.

Db2 cluster services includes three major components:

- Cluster manager: Tivoli SA MP, which includes Reliable Scalable Cluster Technology (RSCT)

- Shared clustered file system: IBM General Parallel File System (GPFS)
- Db2 cluster administration: Db2 commands and administration views for managing and monitoring the cluster

Figure 6. Db2 Cluster services



Db2 cluster services provide essential infrastructure for the shared data instance to be highly available and to provide automatic failover and restart as soon as the instance has been created.

Db2 cluster elements are representations of entities that are monitored and whose status changes are managed by Db2 cluster services. There are three types of Db2 cluster elements:

- Hosts: A host can be a physical machine, LPAR (Logical Partition of a physical machine), or a virtual machine.
- Db2 members: A Db2 member is the core processing engine and normally resides on its home host.
- The home host of a Db2 member is the host name that was provided as the member's location when the member was added to the Db2 shared data instance. A Db2 member has single home host. Db2 members can accept client connections only when they are running on their home host.
- Cluster caching facilities (CFs): The cluster caching facility (CF) is a software application managed by Db2 cluster services that provides internal operational

services for a Db2 shared data instance. There is not necessarily a one-to-one mapping between Db2 cluster elements and the underlying cluster manager resources and resource groups.

Understanding how the Db2 pureScale Feature automatically handles failure

When a failure occurs in the Db2 pureScale instance, Db2 cluster services automatically attempts to restart the failed resources. When and where the restart occurs depends on different factors, such as the type of resource that failed and the point in the resource life cycle at which the failure occurred.

If a software or hardware failure on a host causes a Db2 member to fail, Db2 cluster services automatically restarts the member. Db2 members can be restarted on either the same host (local restart) or if that fails, on a different host (member restart in restart light mode). Restarting a member on another host is called failover.

Member restart includes restarting failed Db2 processes and performing member crash recovery (undoing or reapplying log transactions) to roll back any *inflight transactions* and to free any locks held by them. Member restart also ensures that updated pages have been written to the CF. When a member is restarted on a different host in restart light mode, minimal resources are used on the new host (which is the home host of another Db2 member). A member running in restart light mode does not process new transactions, because its sole purpose is to perform member crash recovery.

The databases on the failed member are recovered to a point of consistency as quickly as possible. This enables other active members to access and change database objects that were locked by the abnormally terminated member. All inflight transactions from the failed member are rolled back and all locks that were held at the time of the abnormal termination of the member are released. Although the member does not accept new transactions, it remains available for resolution of indoubt transactions.

When a Db2 member has failed over to a new host, the total processing capability of the whole cluster is reduced temporarily. When the home host is active and available again, the Db2 member automatically fails back to the home host, and the Db2 member is restarted on its home host. The cluster's processing capability is restored as soon as the Db2 member has failed back and restarted on its home host. Transactions on all other Db2 members are not affected during the failback process.

uDAPL connectivity in Db2 pureScale environments

In a Db2 pureScale environment, the RDMA is supported through the User Direct Access Programming Library (uDAPL) interface.

Similar to TCP/IP, a uDAPL connection request requires at least a target IP address and a port value. The target IP address is the target network address of the connection request that is sent. The port value is for specifying which service (the server application) within the target network location that the connection request is intended for. Therefore, the port number must be unique for each network interface. For Db2 pureScale environments, if there are multiple Host Channel Adapters (HCAs) defined for the CF, the same port value is used for all the interfaces. The port value is the uDAPL connection qualifier that associates a connection request to a specific server application.

The target IP address is specified using either a name format or an IPv4 format. If a name format is used, the name to IP address resolution must be available by using a local /etc/hosts file or by using a DNS server. Typically in a Db2 pureScale installation, a local /etc/hosts file is used to provide the mapping between network names and IP addresses.

Note: The IPv6 address format is not supported.

When the CF is started, the port values are automatically retrieved from the /etc/services file at the CF host. The CF port entries in the /etc/services file use the following convention:

```
DB2CF_<instanceName>_MGMT     <mgmnt_port>/tcp
DB2CF_<instanceName>          <port>/tcp
```

The <*instance_name*> is the name of the Db2 pureScale instance. The <*port*> is the port value that is used for the uDAPL connection. The <*mgmnt_port*> is the port value that is used for the TCP/IP management port connectivity.

Note: During instance creation, unused port values are detected and required entries are created in the /etc/services file to reserve the port values for the Db2 instance.

After you issue the db2start command, the Db2 member attempts to make a number of uDAPL connections to the CF. The member determines the target CF address (the IP address) by using the network names (netnames) associated with the CF entry in the db2nodes.cfg file. If there are multiple HCAs used with the CF, there are multiple netnames for the same CF entry in the db2nodes.cfg file.

From the member host, the CF server port value to use on the connect request is retrieved from the local /etc/services file. This value must match the one used to start CF for the connection to work successfully. It is important to ensure that the /etc/services file content is identical for all the hosts in the Db2 pureScale cluster. Checking the file content prevents a situation where the member and the CF arrive at a different port value after the port name resolution.

The uDAPL configuration information is maintained in the /etc/dat.conf file on each host. It is important to verify that the uDAPL interface is configured correctly on each host. On RHEL, the dat.conf file is located under /etc/ofed for RHEL 5.*x* and /etc/rdma for RHEL 6.*x*.

Manual trace and log file collection

Trace and log files contain information that help troubleshoot problems. If you cannot collect diagnostic data for the IBM Db2 pureScale Feature by using the **db2support** command, you can collect trace and log information manually.

If you can use the **db2support** command with the **-install** parameter, you do not need to use the information in this topic to collect diagnostic information manually. You can manually collect diagnostic information that is similar to what you can collect by using the **db2support** command with the **-install** parameter, but collecting information manually requires more effort.

IBM Support requires you to submit the following files if they are available. Create a single .zip file or .tar archive of all of these files.

- Files from the /tmp directory:
  - Current installation or instance log and trace files in the /tmp directory on the installation-initiating host (IIH). To ensure that the trace and log files that you

collect are relevant to the Db2 installation and configuration commands that returned errors, reissue any commands that failed with the trace option turned on (typically by including the **-d** or the **-t** parameter). For example, if you reissue the **db2icrt** command, include the **-d** command parameter to turn the trace on. The trace output file is named `/tmp/db2icrt.trc.`*`pid`*, where *pid* is a process identifier.

  – Other trace and log files in the `/tmp` directory:
    - The `/tmp/db2trc.trc.*` files
    - The files in the `/tmp/db2log.`*`pid`* directory on the IIH
    - The `/tmp/ibm.db2.cluster.*` files that were generated on the IIH during installation
    - The files in the `/tmp/db2log` directory on all remote hosts
- Files from the *`$HOME`*`/sqllib` directory:
  – The *`$HOME`*`/sqllib/db2nodes.cfg` file
  – The *`$HOME`*`/sqllib/db2dump/db2diag.log` file

  If the *`$HOME`*`/sqllib` directory does not exist, collect the `/tmp/db2nodes.cfg*` and `/tmp/db2diag.log*` files.
- The `/etc/services` file from each host.
- For each node, the errors and trace logs for IBM Tivoli System Automation for Multiplatforms (Tivoli SA MP) from the `/var/ct/`*`db2domainname`* directories and the error and trace logs from the `/tmp/db2_cluster_manager_spooling/` trace spooling directory.[1]
- Output of the following commands:
  – `db2instance -list`, issued as the instance owner
  – `lssam > /tmp/lssam.out`
  – `lslpp -l all | grep "sam.* > /tmp/lslpp_sam.out`
  – `lslpp -l all | grep gpfs > /tmp/lslpp_gpfs.out`
  – `lslpp -l all | grep rsct > /tmp/lslpp_rsct.out`
  – `lsrsrc IBM.Equivalency > /tmp/lsrsrc_IBM.Equivalency.out`
  – `lsrsrc IBM.Application > /tmp/lsrsrc_IBM.Application.out`
  – `db2greg -dump`, from all hosts
  – `ps -fe | grep db2` and `ipcs`, from all hosts
  – `id`, issued as the instance owner and the fenced user on all hosts
  – `uname -a`, from all hosts
  – `errpt -a`, from all hosts
- GPFS cluster file system logs from the `/var/adm/ras/` directory.
- Every file that is located within each node's `/var/ct/iw/` directory and `/var/ct/cfg/` directory.
- A `.zip` file or `.tar` archive of all files in the instance owner's `db2dump` directory (`$HOME/sqllib/db2dump`).

Installation, instance creation and rollback

The Db2 installer installs the Db2 binaries, creates the instance across all hosts (setting up the GPFS cluster and the RSCT peer domain on all Db2 members and CFs), and optionally creates users and groups.

1. For more information on Tivoli SA MP commands, see this https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Documentation%20Central/page/Tivoli%20System%20Automation%20for%20Multiplatforms.

This section contains information that will help you understand and resolve problems that you may encounter during the installation, instance creation and rollback process in a Db2 pureScale environment.

The Db2 Setup wizard does not launch:

If the Db2 Setup wizard fails to launch, consult the error log found in the `DB2DIR/tmp` directory, where *DB2DIR* represents the installation location of your Db2 copy.

Possible reasons for the Db2 Setup wizard not launching include:
- Your system does not meet the installation prerequisites. Review the installation prerequisites and address any problems.
- You do not have an XServer running on the X Window client machine.
- You might not have correctly exported your display. Export your display using the following command:

  ```
  export DISPLAY=IP_Address:0.0
  ```

  where `IP_Address` represents the IP address of the X Window client machine you are using to launch the installation.

Collecting diagnostic data for installation and instance-creation problems:

You might have to collect diagnostic data to solve problems that you encounter when installing Db2 software or creating an instance. If the Db2 installer generates the DBI2047E error message, informing you that installation or instance creation failed, you can use the **db2support** command to automatically collect diagnostic data.

Before you begin

To help ensure that you collect diagnostic data from the largest possible number of sources, you must have root authority. If you issue the **db2support** command with non-root authority, diagnostic data from some sources might not be collected.

To gather information about remote hosts with the **-host** parameter, you require an SSH connection.

About this task

The information that you gather by using the **db2support -install** command might be required by IBM Support to perform analysis. Diagnostic data is collected by issuing the **db2support** command both from log files on disk and from the output of other diagnostic commands.

The **db2support** command that is shipped with Db2 installation images supports only a subset of the command parameters that are available after you install the Db2 product. Until you have installed the Db2 product, the only **db2support** command parameters that you can use are the **-install** and **-host** parameters.

Procedure

To gather diagnostic data:
1. Use the information in the DBI2047E error message to determine the host or hosts where the failure occurred.

2. Issue the **db2support** command in one of the following ways:
   - To collect diagnostic data on the local host, issue the following command:
     ```
     db2support -install
     ```
   - To collect diagnostic data on a remote host, issue the following command:
     ```
     db2support -install -host hostname
     ```
     where *hostname* is the name of the remote host for which you want to collect diagnostic data. For example, to collect diagnostic data on the host hotellnx96, issue the following command:
     ```
     db2support -install -host hotellnx96
     ```
   - To collect diagnostic data on multiple hosts, issue the following command:
     ```
     db2support -install -host hostname_list
     ```
     where *hostname_list* is a comma-separated list of hosts for which you want to collect diagnostic data. For example, to collect diagnostic data on the hosts hotellnx96, hotellnx97, and hotellnx98, issue the following command:
     ```
     db2support -install -host hotellnx96,hotellnx97,hotellnx98
     ```

Results

The diagnostic data is collected in the `db2support.zip` file. The file is created in the current directory, if the current directory is writeable; otherwise, the file is placed in your home directory.

Example

The following example shows the typical output of the **db2support -install** command. In this case, diagnostic data is collected on the local host.

```
              ______  D B 2  S u p p o r t  _____


Output file is "/home/hotellnx96/db2docs1/db2support.zip"
Time and date of this collection: 2010-11-01-10.06.16.559562
Creating directory /home/hotellnx96/db2docs1/DB2SUPPORT
Starting collection on host hotellnx96
 Creating directory /home/hotellnx96/db2docs1/DB2SUPPORT/hotellnx96_IIH
Collecting resources group information
Collecting user identity information
Collecting current process information
Collecting active interprocess communications facilities information
Collecting system information
Collecting detailed data on system and hardware errors
Collecting registry contents
Collecting GPFS information
Collecting configuration, log and trace information for RSCT
Collecting information about installed Db2 products
Collecting information about state of db2 instance
Collecting "/home/hotellnx96/db2docs1/sqllib/db2dump/db2diag.log"
Collecting /etc/services
Collecting /home/hotellnx96/db2docs1/sqllib/db2nodes.cfg
Collecting directory /home/hotellnx96/db2docs1/sqllib/db2dump
Collection on host hotellnx96 completed
Compressing files in /home/hotellnx96/db2docs1/DB2SUPPORT directory
Collecting /home/hotellnx96/db2docs1/db2support.log
Collecting db2support.out


db2support is now complete.
 An archive file has been produced: "/home/hotellnx96/db2docs1/db2support.zip"
```

What to do next

If you are working with IBM Support to troubleshoot a Db2 installation or instance-creation problem, you might be given instructions on how to upload the `db2support.zip` file for analysis by support personnel.

Creating instance resources returns an error:

This example scenario involves a failure while creating the Db2 instance and cluster services resources as part of the installation process. The objective is to determine the problem and diagnose the cause of the failure.

Symptoms

The initial symptom is the following error that is returned while in the step of creating the instance as part of the install.

```
Creating resources for the instance "db2inst1" has failed.
There was an error with one of the issued cluster manager commands. Refer to
the db2diag log file and the Db2 Knowledge Center for details.
```

Diagnosis / resolution

- Check the `/tmp/db2diag.log` for messages similar to the following
  - `Line # : 6884---2610-403 The resource is stale. or`
  - `Line # : 9578---2610-422 Cannot execute the command on node <hostname>. The resource manager IBM.RecoveryRM is not available.`

    Note: If you see these errors, this indicates that the IBM Tivoli System Automation for Multiplatforms (SA MP) recovery resource manager daemon experienced a problem. The daemon serves as the decision engine for Tivoli SA MP and is identified as IBM.RecoveryRM in the system. Diagnostic data will be written by Tivoli SA MP to diagnose the problem.
- Tivoli SA MP diagnostic data is written into the directories `/var/ct/db2domain/log/mc/` (error logs) and `/var/ct/db2domain/run/mc/` (core dumps) and `/tmp/db2_cluster_manager_spooling/` (default trace directory).
- IBM service and development teams use trace and core files for troubleshooting. If you would need IBM service to analyze the diagnostic data, gather the data listed in "Manual trace and log file collection" on page 177.
- Follow these instructions to upload data to IBM Technical Support:
  - Submitting diagnostic information to IBM Technical Support for problem determination
- The IBM Technical Support website is a good source of information, where you can identify known problems based on symptoms or error log messages
  - Db2 support.

Investigating Tivoli SA MP states with Db2 commands:

This topic details how to map the states shown by Tivoli SA MP and RSCT to the states returned by Db2 tools.

Scenarios

Commands from Tivoli SA MP and RSCT which show information can be safely run, however, it is not recommended to run commands that change a state or cause actions to occur in Tivoli SA MP or RSCT. Db2 cluster services controls these resources via the policies setup, and so external changes may result in unexpected actions to occur.

Note: Only the most important mappings are shown in this topic.

The Db2 cluster services tooling commands that show this information are:
- `db2cluster -list -alert`
- `db2instance -list`

When running the `lssam` command and the following scenario is encountered:
- **A resource is failed offline**
  - A failed offline resource may not indicate a problem, based on what is specified in the output. To see if this maps to a problem, run the following commands:
  -
    1. `db2instance -list`
    2. `db2cluster -list -alert`
  - Run the `db2cluster -list -alert` command and follow the instructions returned by that command. If you are unable to resolve the issue, follow the instructions that follow:
    - If a `db2start` command was issued, see "CF server failure" on page 198.
    - If a `db2start` command was not issued, contact IBM Service.
  - Failed offline states may be cleared via the `db2cluster -clear -alert` command if the alert appears in the `db2cluster -list -alert` command output. It is not recommended to clear these states via Tivoli SA MP or RSCT commands.
- **A resource is pending online**
  - Wait for the resource to come online if a `db2start` command was issued, or if the resource is undergoing a restart operation.
    - If the state later moves to failed offline see **A resource is failed offline** .
  - Run `db2instance -list` to see whether or not the instance has been started. Pending Online states may appear in lssam output when the instance is stopped, and is waiting for `db2start` to be issued by the user.
- **A resource is offline**
  - `db2stop` has been issued and has completed.
  - `db2start` has been issued, a dependent resource may be unavailable. Issue `db2cluster -list -alert` to see if there are any alerts that may need to be cleared, or investigated. For instance, a network adapter may be malfunctioning, or a host may be unavailable.
- **A resource is pending offline**
  - If `db2stop` was issued, wait for the resource to change to offline.
  - If `db2stop` was not issued, this could be the result of a member returning to it's home host. Run `db2instance -list` to verify that it comes up again. Run `db2cluster -list -alert` to verify that there are no issues that may have caused the Pending Offline to occur. See the section on '**A resource is offline** if the pending offline transitions to offline.

2. For more information on Tivoli SA MP commands, see this https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Documentation%20Central/page/Tivoli%20System%20Automation%20for%20Multiplatforms.

- **A resource is unknown**
  - This is a transient state. Rerun **lssam** to see if the state clears itself.
  - A machine may be offline, resulting in the inability to run the monitor.
  - run **db2instance -list** to see whether the host is active. If the host is not active, contact IBM Service.
- **A resource group has an ineligible state**
  - This may be a transient state. Rerun **lssam** to see if the state clears itself.
    - If the problem persists then run **db2instance -list** and **db2cluster -list -alert** for more information about what steps to take next.

Creating instance resources hangs the installation:

This example scenario involves a hang while creating the Db2 instance and cluster services resources as part of the installation process. The objective is to determine the problem and diagnose the cause of the hang. There are two cases for this scenario.

Case 1: Messages exist in the install log or the db2diag log file

Symptoms

The initial symptom in this case is a hang during the installation process. More specifically, the hang occurs during the process of creating the instance.

Diagnosis and resolution:

- Check the `/tmp/db2setup.log` and/or the `/tmp/db2icrt.log`. For this example, the following message exists

  ```
  Creating resources for the instance "db2inst1" has failed.
  There was an error with one of the issued cluster manager commands.
  Refer to the db2diag log file and the Db2 Knowledge Center for details.
  ```

  Check to see if you have a similar message.
- Check the `/tmp/db2diag.log` for messages similar to the following ones:
  - `Line # : 6884---2610-403 The resource is stale.` or
  - `Line # : 9578---2610-422 Cannot execute the command on node <hostname>. The resource manager IBM.RecoveryRM is not available.`
- If you see these errors, this indicates the IBM Tivoli System Automation for Multiplatforms (SA MP) recovery resource manager daemon experienced a problem. The daemon serves as the decision engine for Tivoli SA MP and is identified as IBM.RecoveryRM in the system. Diagnostic data will be written by Tivoli SA MP to diagnose the problem.
- Tivoli SA MP diagnostic data is written into the directories `/var/ct/db2domain/log/mc/` (for error logs) and `/var/ct/db2domain/run/mc/` (core dumps) and `/tmp/db2_cluster_manager_spooling/` (default trace directory).
- IBM service and development teams use trace and core files for troubleshooting. If you would like IBM service to analyze the diagnostic data, gather the data listed under the topic "Manual trace and log file collection" on page 177
- Follow these instructions to upload data to IBM Technical Support:
  - Submitting diagnostic information to IBM Technical Support for problem determination
- The IBM Technical Support website is a good source of information, where you can identify known problems based on symptoms or error log messages

– Db2 support.

Case 2: No errors or messages in the install log or the db2diag log file

Symptoms

- The initial symptom is a hang during the installation process.
- The state of the hang is such that there might not be any messages reported into the `/tmp/db2setup.log install log`, or the log does not exist

Diagnosis and resolution:

- If the `/tmp/db2setup.log` and/or `/tmp/db2icrt.log exist`, check if you have a similar message to:
  - `Creating resources for the instance "db2inst1" has failed.`
  - `There was an error with one of the issued cluster manager commands. Refer to the db2diag log file and the Db2 Information Center for details.`
- If the `/tmp/db2diag.log` exists check for messages similar to the following ones:
  - `Line # : 6884---2610-403 The resource is stale. or`
  - `Line # : 9578---2610-422 Cannot execute the command on node <hostname>. The resource manager IBM.RecoveryRM is not available.`
- If you see these errors, this indicates the Tivoli SA MP recovery resource manager daemon experienced a problem. The daemon serves as the decision engine for Tivoli SA MP and is identified as IBM.RecoveryRM in the system. Diagnostic data will be written by Tivoli SA MP to diagnose the problem.
- Tivoli SA MP diagnostic data is written into the directories `/var/ct/db2domain/log/mc/` (for error logs) and `/var/ct/db2domain/run/mc/` (core dumps) and `/tmp/db2_cluster_manager_spooling/` (default trace directory).
- If the `/tmp/db2setup.log` or `/tmp/db2diag.log` files do not exist or are empty, gather as much of the remaining data listed under the topic "Manual trace and log file collection" on page 177 as possible. Contact IBM service for assistance.
- Follow these instructions to upload data to IBM Technical Support:
  - Submitting diagnostic information to IBM Technical Support for problem determination

Manually update IBM Spectrum Scale to meet Db2 pureScale Feature requirements (AIX):

IBM Spectrum Scale is installed as part of Db2 pureScale Feature installation. However, during a Db2 pureScale Feature installation, if a IBM Spectrum Scale cluster already exists but was not created by the Db2 installer, or, you manually updated the IBM Spectrum Scale level, a failure can occur when you install the Db2 pureScale Feature. A failure occurs if the already installed IBM Spectrum Scale level does not match the release or efix level that is required by the Db2 installer.

Before you begin

- If an efix is required, make the IBM Spectrum Scale efix available to all IBM Spectrum Scale hosts on a local or Network File System (NFS). The fix has the file extension *.epkg.Z and is located on the Db2 pureScale Feature installation image in the following directory: `<`*`DB2-image-directory`*`>/db2/aix/gpfs/efix`

About this task

IBM Spectrum Scale is currently the only supported file system in a Db2 pureScale environment. Prior to installing Db2 pureScale Feature, if a IBM Spectrum Scale cluster is not already on your system, the Db2 installer automatically installs IBM Spectrum Scale, and creates a IBM Spectrum Scale cluster and filesystem. When the Db2 installer creates a IBM Spectrum Scale cluster and filesystem, this is referred to as a “Db2 managed” cluster. However, if a IBM Spectrum Scale cluster already exists on your system, this is referred to as a “user-managed” cluster.

After a Db2 managed cluster is installed on your system, you must not manually update the IBM Spectrum Scale level, or manually apply a non-security related efix. If you do manually update the IBM Spectrum Scale level, or manually apply a non-security efix, problems can occur when you upgrade to a new Db2 version. (The problem occurs when you attempt to install the new Db2 version.) You receive an error message indicating IBM Spectrum Scale is already installed and you must manually update your IBM Spectrum Scale cluster. If this error occurs, before you can continue, you must follow the steps in this topic.

For instructions on manually updating a security efix, please refer to the "Online upgrade of IBM Spectrum Scale level with a security-related efix" topic. When upgrading to a new Db2 version at a later time, you may receive an error message indicating IBM Spectrum Scale is already installed and you must manually update your IBM Spectrum Scale cluster. If this error occurs, before you can continue, you must follow the steps in this topic.

Procedure

Perform these steps on one host at a time, until all hosts are updated.

Note: To avoid cluster failover and reelecting a cluster manager, update the cluster manager last.

1. Log on as root.
2. Compare the existing IBM Spectrum Scale release and efix level already installed, to the levels on the installation media.

   To verify the IBM Spectrum Scale release already installed on the system, run:

   ```
   <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -v install
   ```

   To verify the IBM Spectrum Scale release on the installation media, run:

   ```
   <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -v media
   ```

   If the IBM Spectrum Scale release level already installed is at a lower level than the level on the installation media, a IBM Spectrum Scale release level update is required. Otherwise, continue to verify the IBM Spectrum Scale efix level.

   To verify a IBM Spectrum Scale efix level already installed on the system, run:

   ```
   <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -s install
   ```

   To verify the IBM Spectrum Scale efix level on the installation media, run:

   ```
   <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -s media
   ```

   If the IBM Spectrum Scale efix level already installed is at a lower level than the level on the installation media, a IBM Spectrum Scale modification level update.

3. The remaining steps must be performed one host at a time on all hosts.

   Stop IBM Spectrum Scale on the current host:

   ```
   /usr/lpp/mmfs/bin/mmshutdown
   ```
4. Verify that IBM Spectrum Scale has shutdown properly. The IBM Spectrum Scale state on the host must indicate "down".

   ```
   /usr/lpp/mmfs/bin/mmgetstate -a
   ```
5. Unload the Monitoring Multimedia File System (MMFS) kernel extension:

   ```
   /usr/lpp/mmfs/bin/mmfsenv -u
   ```

   Note: If the command indicates that kernel extensions are "busy", to ensure that the updated kernel extension is loaded, the host must be rebooted after upgrading IBM Spectrum Scale. Alternatively, identify and kill processes that have a current working directory or open file handles on the IBM Spectrum Scale cluster. If after terminating the processes, and rerunning the command no longer indicates a "busy" state, a reboot can be avoided.
6. Update either the IBM Spectrum Scale release level or IBM Spectrum Scale modification level. Perform one of these steps:
   - To update the IBM Spectrum Scale release level (for example, from 3.3 to 3.4), run:

     ```
     <DB2-image-directory>/db2/aix/gpfs/installGPFS -g
     ```
   - To update the IBM Spectrum Scale modification level (for example, from 3.3.1 to 3.3.4), run:

     ```
     <DB2-image-directory>/db2/aix/gpfs/installGPFS -u
     ```
7. Start IBM Spectrum Scale on the host:

   ```
   /usr/lpp/mmfs/bin/mmstartup
   ```
8. Verify that IBM Spectrum Scale has started properly. The IBM Spectrum Scale state on the host must indicate "active".

   ```
   /usr/lpp/mmfs/bin/mmgetstate -a
   ```
9. Mount all IBM Spectrum Scale file systems:

   ```
   /usr/lpp/mmfs/bin/mmmount all
   ```
10. Verify the IBM Spectrum Scale level now matches the release or efix level that is required by the Db2 installer.

    Verify the IBM Spectrum Scale release installed on the system matches the IBM Spectrum Scale release on the installation media:

    ```
    <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -v install
    <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -v media
    ```

    Verify a IBM Spectrum Scale efix level installed on the system matches the IBM Spectrum Scale efix level on the installation media:

    ```
    <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -s install
    <DB2-image-directory>/db2/aix/gpfs/db2ckgpfs -s media
    ```
11. Repeat steps 3 to 10 on the next host until all hosts are updated.

Results

All IBM Spectrum Scale hosts are now at the required code level for Db2 pureScale Feature.

What to do next

Continue installing Db2 pureScale Feature.

Manually update IBM Spectrum Scale to meet Db2 pureScale Feature requirements (Linux):

IBM Spectrum Scale is installed as part of Db2 pureScale Feature installation. However, during a Db2 pureScale Feature installation, if a IBM Spectrum Scale cluster already exists but was not created by the Db2 installer, or, you manually updated the IBM Spectrum Scale level, a failure can occur when you install the Db2 pureScale Feature. A failure occurs if the already installed IBM Spectrum Scale level does not match the release or efix level that is required by the Db2 installer.

About this task

IBM Spectrum Scale is currently the only supported file system in a Db2 pureScale environment. Prior to installing Db2 pureScale Feature, if a IBM Spectrum Scale cluster is not already on your system, the Db2 installer automatically installs IBM Spectrum Scale, and creates a IBM Spectrum Scale cluster and filesystem. When the Db2 installer creates a IBM Spectrum Scale cluster and filesystem, this is referred to as a "Db2 managed" cluster. However, if a IBM Spectrum Scale cluster already exists on your system, this is referred to as a "user-managed" cluster.

After a Db2 managed cluster is installed on your system, you must not manually update the IBM Spectrum Scale level, or manually apply a non-security related efix. If you do manually update the IBM Spectrum Scale level, or manually apply a non-security efix, problems can occur when you upgrade to a new Db2 version. (The problem occurs when you attempt to install the new Db2 version.) You receive an error message indicating IBM Spectrum Scale is already installed and you must manually update your IBM Spectrum Scale cluster. If this error occurs, before you can continue, you must follow the steps in this topic.

For instructions on manually updating a security efix, please refer to the "Online upgrade of IBM Spectrum Scale level with a security-related efix" topic. When upgrading to a new Db2 version at a later time, you may receive an error message indicating IBM Spectrum Scale is already installed and you must manually update your IBM Spectrum Scale cluster. If this error occurs, before you can continue, you must follow the steps in this topic.

Procedure

To update IBM Spectrum Scale to meet Db2 pureScale Feature requirements:

1. Log on as root.
2. Compare the existing IBM Spectrum Scale release and efix level already installed, to the levels on the installation media.

   To verify the IBM Spectrum Scale release already installed on the system, run:

   ```
   <DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -v install
   ```

   To verify the IBM Spectrum Scale release on the installation media, run:

   ```
   <DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -v media
   ```

   If the IBM Spectrum Scale release level already installed is at a lower level than the level on the installation media, a IBM Spectrum Scale release level update is required. Otherwise, continue to verify the IBM Spectrum Scale efix level.

   To verify the IBM Spectrum Scaleefix level already installed on the system,

   ```
   <DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -s install
   ```

To verify the IBM Spectrum Scale efix level on the installation media, run:

```
<DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -s media
```

If the IBM Spectrum Scale efix level already installed is at a lower level than the level on the installation media, a IBM Spectrum Scale modification level update.

3. Stop IBM Spectrum Scale on the current host:
   ```
   /usr/lpp/mmfs/bin/mmshutdown
   ```
4. Verify that IBM Spectrum Scale has shutdown properly: The IBM Spectrum Scale state on the host must indicate "down".
   ```
   /usr/lpp/mmfs/bin/mmgetstate -a
   ```
5. Update either the IBM Spectrum Scale release level or IBM Spectrum Scale modification level. Perform one of these steps:
   - To update the IBM Spectrum Scale release level (for example, from 3.3 to 3.4), run:
     ```
     <DB2-image-directory>/db2/linuxamd64/gpfs/installGPFS -g
     ```
   - To update the IBM Spectrum Scale modification level (for example, from 3.3.1 to 3.3.4), run:
     ```
     <DB2-image-directory>/db2/linuxamd64/gpfs/installGPFS -u
     ```
6. After updating either the IBM Spectrum Scale release level or IBM Spectrum Scale modification level, you must build the GPFS portability layer by issuing these commands:
   ```
   cd /usr/lpp/mmfs/src
   make Autoconfig
   make World
   make InstallImages
   ```
7. Start IBM Spectrum Scale on the host:
   ```
   /usr/lpp/mmfs/bin/mmstartup
   ```
8. Verify that IBM Spectrum Scale has started properly. The IBM Spectrum Scale state on the host must indicate "active".
   ```
   /usr/lpp/mmfs/bin/mmgetstate -a
   ```
9. Mount all IBM Spectrum Scale file systems:
   ```
   /usr/lpp/mmfs/bin/mmmount all
   ```
10. Verify the IBM Spectrum Scale level now matches the release or efix level that is required by the Db2 installer.

    Verify the IBM Spectrum Scale release installed on the system matches the IBM Spectrum Scale release on the installation media:
    ```
    <DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -v install
    <DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -v media
    ```
    Verify a IBM Spectrum Scale efix level installed on the system matches the IBM Spectrum Scale efix level on the installation media:
    ```
    <DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -s install
    <DB2-image-directory>/db2/linuxamd64/gpfs/db2ckgpfs -s media
    ```
11. Repeat steps 3 to 10 on the next host until all hosts are updated.

Results

All IBM Spectrum Scale hosts are now at the required code level.

What to do next

Continue installing Db2 pureScale Feature.

Validating a GDPC through testing:

Geographically dispersed Db2 pureScale cluster (GDPC) ensures that the cluster is running if a total site failure were to happen at a site. There are multiple tests that can be run to ensure that GDPC is running properly.

Running these tests before a disaster strikes will ensure that the geographically dispersed Db2 pureScale cluster (GDPC) has been correctly configured, and help with a hands-on understanding of the expected behavior in each failure case:

1. Soft member failure – kill -9 of the **db2sysc** process.
2. Soft cluster cache facility (CF) failure, primary or secondary – kill -9 of the **ca-server** process.
3. Hard member failure – reboot of the member LPAR.
4. Hard CF failure, primary or secondary – reboot of the CF LPAR.
5. Storage shutdown at one site.
6. Shutdown of the tiebreaker site.
7. Pull the cable for the private network over where member-to-CF communicate (either over IB, or using RoCE or TCP/IP) from one site.
8. Shutdown the IB or RoCE switch at one site.
9. Pull the Ethernet cable at one site.
10. If you are using IB, shutdown the longbow at one site.
11. Site failure – shutdown the LPARs, switches, Longbow, and storage at one site to simulate a total site failure.

Note that there are variations of site failure. The type of failure depends on whether the failed site is the IBM Spectrum Scale cluster manager, the file system manager, the RSCT group leader, the TSA master, or if it is the primary or secondary CF

Frequently asked questions about Db2 pureScale Feature host validation problems:

The following sections provide possible solutions to problems you might encounter when attempting to validate remote hosts.

What if the user id specified is invalid?

The Db2 installer validates the existing user id specified. If the user id specified is invalid, ensure the user id exists on the installation-initiating host (IIH) and on each of the remote hosts. Ensure the user id has the exact same uid, gid, group name, and directory path. To create a user id, see the "Required users for a Db2 pureScale Feature installation" topic.

What if the ports specified are invalid?

Before instance creation, the Db2 installer validates the ports you specified. If the ports you specified are not available, review the TCP/IP ports that are already in use the IIH and on the remote hosts by opening the `/etc/services` file. After you have determined a range of free ports, enter the new ports into the applicable ports fields in the Advanced Settings Panel of the Add Host Panel and have them validated by the Db2 installer.

What if a host already belongs to another GPFS cluster?

A host cannot be a member of two GPFS clusters. If the IIH is not part of a GPFS cluster, then the remote host should not belong to a GPFS cluster. If the IIH is part of a GPFS cluster, then the remote host should either not belong to a GPFS cluster or belong to the same GPFS cluster. After you have resolved this, try adding the host to the host list again.

What if the global registry variable record on the host indicates a GPFS Cluster already exists?

In some cases, the clean up of the global registry might not have been complete, leaving behind a record (GPFS_CLUSTER) that indicates there is a GPFS cluster in use by Db2 on the host when in fact there is not one. Contact IBM Software Support.

What if the IIH cannot connect with the target host?

This error can occur as the Db2 installer tries to verifying ssh communication from the IIH to the target host. Ensure that the settings for ssh and scp are correctly setup. ssh and scp need to be setup without a password prompt between the hosts for the root user. For more information, see the "Installing and setting up OpenSSH on AIX" topic.

If the problem occurs outside the Db2 installer, you can check a variety of items to determine the source of the problem. For example, this communication failure could result from a bad physical connection, a defective or unresponsive network adapter driver, or a misconfigured network.

What if the attempt to communicate with a target host times out?

If an attempt to validate a host did not finish within the time out period, it will time out. Check the connections to the host and try to add it again. You can also change the time out variable and re-run the installation command.

What if the Db2 installer detects an existing RSCT peer domain on a host?

During a Db2 installation, only one RSCT peer domain can be active at a time. The peer domain that was not created by the Db2 installer must be stopped or removed before creating a RSCT peer domain to be used by the IBM Db2 pureScale Feature.

To stop the RSCT peer domain using the db2cluster command on the host db2host1, log on to a host that belongs to the same active RSCT peer domain, run the `db2cluster -cm -stop -host db2host1` command. If the **`db2cluster`** command is not available, run the `stoprpdomain <domain name>` command. Run the **`lsrpdomain`** command to determine the domain name to specify.

If an attempt to validate host did not finish within the time out period, it will time out. Check the connections to the host and try to add it again. You can also change the time out variable and re-run the installation command.

To remove the RSCT peer domain:
1. Remove the remote host from the host list.

2. If the remote host "db2host1" belongs to a different Db2 pureScale instance, remove it from that Db2 pureScale instance using the "db2iupdt -drop" command.

To remove the host from a Db2 pureScale instance:
1. Log on to a different host which belongs to the same Db2 pureScale instance
2. Run
   ```
   db2iupdt [-d] -add -m|cf <host_name>:<interconnect_name> -u <fenced_id>
   <instance_owner>
   ```

To remove a remote host that does not belong to a Db2 instance, run one of the following commands:
- `db2cluster -cm -remove -host <hostname>`
- `rmrpnode <hostname>`

What if there is a conflict between the DEFAULT_INSTPROF record and the instance shared directory specified?

The Db2 installer has detected a conflict between the DEFAULT_INSTPROF record and the instance shared directory specified. Do not specify the instance shared directory. The DEFAULT_INSTPROF record in the global registry indicates the instance shared file system has already been set up by Db2.

In this case, the following options or keywords are not needed.
- For a response file installation: INSTANCE_SHARED_DEVICE_PATH and INSTANCE_SHARED_DIR.
- For db2icrt or db2iupdt: instance_shared_dev and instance_shared_dir.

If the value for INSTANCE_SHARED_DIR / instance_shared_dir matches with the existing instance shared file system mount point, the Db2 installer will still allow to pass. However, if the value does not match, the installation will fail.

What if the PVID for the device path on the install-initiating host (IIH) does not exist on a target host?

Ensure that PVID is setup on the IIH and all the hosts. See the "Configuring PVIDs for a Db2 pureScale Feature instance" topic for more details.

What if the cluster interconnect netname specified is invalid?

This error occurs because the Db2 installer cannot identify the cluster interconnect netname. Try following methods to solve the problem:
1. Check the command for a typo error.
2. Check the machine's network configuration (for example check /etc/hosts).
3. Use the ping and nslookup tools to confirm the cluster interconnect netname maps to the same IP address on all hosts. Check that IP address maps back to the same cluster interconnect netname on all hosts.

What if the cluster interconnect netname for a target host failed to ping the IIH?

When adding a new host, the Db2 installer checks to see if the new host can send a small data packet to the IIH and receive a reply. This send-reply test is commonly known as a ping - this message is the result of a failed ping. If there is a problem, you can verify the results by running this command line from the remote host's console: `ping IIH address_or_name`.

If this test fails, ensure that the ssh communication between the IIH and the remote hosts is set up correctly.

After it is verified that the problem occurs outside the Db2 installer, there are various things the you can check find the source of the problem, for example bad physical connections (for example, loose cable), a faulty network adapter driver, or an improperly setup network. Check the network adapter and cable, or choose a different one.

What if the cluster interconnect netname for a target host is not on the same subnet as the IIH?

You can reconfigure the network adapter or choose a different one. This error occurs when the cluster interconnect netname for a host is not on the same subnet as the installation-initiating host. All cluster interconnects for CF need to be on the same subnet for performance reasons (all hosts within the same subnet can usually be reached in one routing hop).

For example if the cluster interconnect network is configured with the network address 192.168.0.0/24, then all cluster interconnect netname addresses should start with 192.168.0 (for example, 192.168.0.1, 192.168.0.2, and so on).

Check the network card configuration on the new host (for example, run the `ifconfig -a` command) and check `/etc/hosts` if a name was used instead of an address.

Frequently asked questions about installation, instance creation, and rollback problems with the Db2 pureScale Feature:

Use the answers in this list of frequently asked questions to help you provide possible solutions to problems that might arise during the installation process of the IBM Db2 pureScale Feature.

Ensure that you issue any installation or creation commands with the debug (-d) parameter to generate the trace file which will assist in any debugging attempts.

Where can I find the installation log files?

The most important source of information for troubleshooting installation problems are the installation log files. The Db2 setup log file, `db2setup.log` by default, captures all Db2 installation information including errors. The Db2 error log file, `db2setup.err`, captures any error output that is returned by Java applications (for example, exceptions and trap information).

By default, the log files are located in the *DB2DIR*`/tmp` directory. You might have overridden this default when you issue the **db2setup** command. The Db2 installer copies the Db2 setup log file to the *DB2DIR*`/install/logs/` directory, and renames it `db2install.history`. If there are insufficient details in the log files on the installation-initiating host (IIH), you can also check the host logs that detail the actions on each remote host, and can be found under the *DB2DIR*`/tmp/db2log.`*xxxx* directory (where *xxxx* represents the process id) and *DB2DIR*`/tmp/db2log` directory.

What is a rollback?

A rollback occurs when an operation was unsuccessful and the Db2 installer will clean up the environment. You can have an instance rollback and a Db2 binaries rollback. If the Db2 installer fails to create a Db2 instance, it will trigger a rollback of the instance on all hosts. An instance rollback does not rollback Db2 binaries.

A partial rollback occurs when you can set up an instance, but it cannot be extended to a particular host.

What if the Db2 binary installation fails on one or more hosts?

Check the installation logs on the IIH and if more failure details are required, check the logs on the appropriate remote hosts. Resolve the errors and restart the installation.

What if my instance was not created?

If your instance creation has failed, a rollback of the instance on all involved hosts has occurred. Refer to the installation log for more details. If the Db2 binaries are successfully installed on a host, they will not be rolled back. Resolve the errors outlined in the installation logs and run the appropriate command (**db2isetup** or **db2icrt**) command to create the instance again. You do not need to launch the Db2 installer.

What if my instance was only partially created? What if my instance was created, but some of my members and CFs were not created?

It might occur during an installation that your instance was created only on some of the target hosts. Instance rollback happens on the hosts where the instance creation was not completed without triggering a binary rollback. You will receive a post installation message showing the hosts that have not been included in the instance due to an error.

Resolve any errors outlined in the installation logs. After you've resolved the problem, you can run the **db2isetup** or **db2iupdt** command to add the members or CFs:

- You can validate the host and add it to the instance by using the Db2 Instance Setup wizard (by issuing **db2isetup** command).
- You can extend your instance by issuing the **db2iupdt -add** command from a host that already belongs to the instance.

What if I cannot install the rsct.basic file?

If you have RSCT packages installed that have a higher version number than the provided `rsct.basic` file, you cannot install Tivoli SA MP. To resolve this error, refer to the rsct.basic cannot be installed on AIX support documentation.

Frequently asked questions about Db2 cluster file system problems:

Use the following information to determine possible solutions to problems with the Db2 cluster file system, which is based on IBM Spectrum Scale.

What if the IBM Spectrum Scale cluster failed to be created?

During instance creation, the IBM Spectrum Scale cluster might fail to be created. This failure will trigger an instance creation rollback. Review the log file found in the `DB2DIR/tmp` directory. The **`db2cluster`** command is used to create an instance. The **`db2cluster`** command log file can be found in `DB2DIR/tmp/ibm.db2.cluster.*`. After reviewing the contents of that log file, contact IBM Software Support.

What if the IBM Spectrum Scale cluster fails to extend to the target host?

If the IBM Spectrum Scale cluster extension to the host fails, you can review the log file, found in the `DB2DIR/tmp` directory to determine why the extension was prevented on that host.

What if the "GPFS_CLUSTER" global registry variable record on the target hosts was not created?

As with any errors, review the installation log for specific details. Ensure that the IBM Spectrum Scale cluster has been properly created. If the log indicates that the GPFS_CLUSTER global registry variable record was not created on the specified hosts, contact IBM Software Support.

What if the IBM Spectrum Scale cluster was not deleted during a failed installation?

If a failed attempt to install a Db2 product results in a rollback, the Db2 installer might be unable to remove the newly created IBM Spectrum Scale file system. To understand why the IBM Spectrum Scale cluster was not removed, review the log file found in the *`DB2DIR`*`/tmp` directory .

GDPC and IBM Spectrum Scale replication FAQ:

This FAQ provides you with answers to common questions about geographically dispersed Db2 pureScale cluster (GDPC) and IBM Spectrum Scale™ replication environment problems.

What do I do when I cannot bring the disks online after a storage failure on one site was rectified?

If nodes come online before the storage device, you must ensure that the disk configurations are defined and available before you try to restart the failed disk. If the **`Device`** and **`DevType`** fields are marked by a `-` when you try to list the network shared disk (NSD) using the `mmlsnsd -X` command, you must ensure that the device configurations for the disk services are available before attempting to restart the disks. Please consult the operating system and device driver manuals for the exact steps to configure the device. On AIX platforms, you can run the **`cfgmgr`** command to automatically configure devices that have been added since the system was last rebooted.

What do I do if a computer's IP address, used for the IB interface, cannot be pinged after a reboot?

Ensure the InfiniBand (IB) related devices are available:

```
root> lsdev -C | grep ib
ib0        Available      IP over InfiniBand Network Interface
iba0       Available      InfiniBand host channel adapter
```

If the devices are not available, bring them online with **chdev**:

```
chdev -l ib0 -a state=up
```

Ensure that the *ib0*, *icm* and *iba0* properties are set correctly, that *ib0* references an IB adapter such as *iba0*, and that properties are persistent across reboots. Use the **-P** option of **chdev** to make changes persistent across reboots.

What do I do if access to the IBM Spectrum Scale file systems hangs for a long time on a storage controller failure?

Ensure the device driver parameters are set properly on each machine in the cluster.

What do I do if the cluster comes down following a site failure?

Check the system logs on the surviving site to see if IBM Spectrum Scale has triggered a kernel panic due to outstanding I/O requests:

```
GPFS Deadman Switch timer has expired and there are still outstanding I/O requests
```

If this is the case, then ensure that the device driver parameters have been properly set

What happens when one site loses Ethernet connectivity and the LPARs on that site are expelled from the IBM Spectrum Scale cluster?

If the IBM Spectrum Scale cluster manager is on the tiebreaker site this behavior is expected, as the cluster manager does not have IB or Remote Direct Memory Access (RDMA) over Converged Ethernet (RoCE) connectivity and can no longer talk to the site which has lost Ethernet connectivity. If the IBM Spectrum Scale cluster manager is not on the tiebreaker, but is on the site that retains Ethernet connectivity then ensure that the tiebreaker site is a IBM Spectrum Scale quorum-client, not a quorum-manager, as per the **mmaddnode** command. If the tiebreaker host is a quorum-manager its status can be changed to client with the `/usr/lpp/mmfs/bin/mmchnode --client -N hostT` command. The status of the nodes as managers or clients can be verified with the `/usr/lpp/mmfs/bin/mmlscluster` command. Also ensure that the IBM Spectrum Scale subnets parameter has been set properly to refer to the IP subnet that uses the IB or RoCE interfaces. The `/usr/lpp/mmfs/bin/mmlsconfig` command can be used to verify that the subnets parameter has been set correctly.

What do I do when one site loses ethernet connectivity and the members on that site remain stuck in STOPPED state instead of doing a restart light and going to WAITING_FOR_FAILBACK state?

Ensure that LSR has been disabled.

How can I remove unused IBM Spectrum Scale Network Shared Disks (NSD)?

Scenario that can lead to the need to manually remove unused NSDs:
1. User-driven or abnormal termination of **db2cluster** CREATE FILESYSTEM or ADD DISK command.
2. The unused NSDs were created manually at some point earlier but left in the system.

The free NSDs need to be removed before they can be used in **db2cluster** command with either CREATE FILESYSTEM or ADD DISK option. Use the following instructions to remove them:

Note: Run all the following commands on the same host.

1. Run **mmlsnsd -XF** to list the free NSD and its corresponding device name.

```
root@coralpib21a:/> mmlsnsd -XF

Disk name    NSD volume ID      Device         Devtype  Node name                Remarks
---------------------------------------------------------------------------------------------------
gpfs2118nsd  09170151FFFFD473   /dev/hdisk7    hdisk    coralpib21a.torolab.ibm.com
```

2. Find the NSD name that matches the target device to be removed.

   `gpfs2118nsd`

3. Run **mmdelnsd <NSD name>** to remove the desired unused NSD.

```
root@coralpib21a:/> mmdelnsd gpfs2118nsd
mmdelnsd: Processing disk gpfs2118nsd
mmdelnsd: Propagating the cluster configuration data to all
affected nodes.  This is an asynchronous process.
```

Frequently asked questions about RSCT peer domain problems:

The following sections provide possible solutions to problems with your RSCT peer domain.

What if the "PEER_DOMAIN" global registry variable record on the target hosts was not created?

As with any errors, review the installation log for specific details. If the log indicates that the PEER_DOMAIN global registry variable record was not created on the specified hosts, contact IBM Software Support.

What if the Tivoli SA MP resource failed to be created?

The Tivoli SA MP resource failed to be created during instance creation. This error has triggered an instance rollback. See the log file to determine the reason why the Tivoli SA MP resource failed.

The log file can be found in `*instance_user*/sqllib/db2dump/db2diag.log`. Because a rollback rolls back the contents *instance_user*/sqllib, the file is backed up as /tmp/db2diag.log.process_id. There are multiple causes for this error. A possible reason for this is that the db2nodes.cfg is incorrect or failed to be created. The file can be found at *instance_user*/sqllib/db2nodes.cfg.

Fix the problem and run the **db2icrt** command. If you are unable to resolve the problem, contact IBM Software Support with all log and trace files.

Frequently asked questions about response file installation problems with the Db2 pureScale Feature

Use this topic to help you resolve response file problems during installation.

Where is the generated response file saved?

By default, the response file generated by the Db2 Setup wizard or the Db2 Instance wizard is saved in the *DB2DIR*/tmp/ directory with the name db2ese_dsf.rsp. *DB2DIR* represents the installation location of your Db2 copy.

You can specify the name and location of the response file in the **Response file name** field of the "Select installation, response file creation, or both" Panel of the Db2 Setup wizard or the Db2 Instance wizard. You should save the response file from a host that will be designated a Db2 member.

Where can I find the IBM Db2 pureScale Feature sample response file?

The Db2 pureScale Feature sample response file, `db2dsf.rsp`, is located in *`DB2DIR`*`/install/db2/`*`platform`*`/samples` directory, where *platform* refers to the appropriate operating system.

Which keywords are mandatory for a Db2 pureScale Feature installation?

The Db2 pureScale Feature release introduced new keywords some of which are mandatory depending on the environment you want to create. Review the list of keywords to determine which keys are mandatory for your environment. See the "Response file keywords" topic for more details.

Post-installation

This section contains information that will help you understand, isolate, and resolve problems that you might encounter after the installation of the IBM Db2 pureScale Feature.

Member start failure:

Use the information in this topic to help you diagnose if a member component failed.

Symptoms

A Db2 instance fails to start on the execution of the **db2start** command.

Diagnosing a member start failure

- Refer to the SQLCODEs in the **db2start** command output.
- The state of each member can be viewed by issuing the **db2instance -list** command. This information might show members in a STOPPED or ERROR state if the startup has failed, depending on when the failure occurs.
  - The following example shows a sample output from the **db2instance -list** command:

```
ID     TYPE   STATE     HOME_HOST    CURRENT_HOST   ...
--     ----   -----     ---------    ------------
0    MEMBER   ERROR     host01       host01
1    MEMBER   STARTED   host02       host02
2    MEMBER   STARTED   host03       host03
128  CF       PRIMARY   host04       host04
129  CF       PEER      host05       host05


ALERT   PARTITION_NUMBER       LOGICAL_PORT    NETNAME   ...
-----   ----------------       ------------    -------
YES                    0                  0    host01-ib0
NO                     0                  0    host02-ib0
NO                     0                  0    host03-ib0
NO                     -                  0    host04-ib0
NO                     -                  0    host05-ib0
```

```
HOSTNAME      STATE      INSTANCE_STOPPED      ALERT
--------      -----      ----------------      -----
host01       ACTIVE                    NO         NO
host02       ACTIVE                    NO         NO
host03       ACTIVE                    NO         NO
host04       ACTIVE                    NO         NO
host05       ACTIVE                    NO         NO
```

- For more information about the listed alerts, run the **db2cluster -cm -list -alert** command. For example, the **db2cluster -cm -list -alert** command might return something like the following alert:

  ```
  Alert: Db2 member '0' failed to start on its home host 'host01'. The
  cluster manager will attempt to restart the Db2 member in restart
  light mode on another host. Check the db2diag log file for messages
  concerning failures on hosts 'host01' for member '0'

  Action:
  This alert must be cleared manually with the command:
  db2cluster -cm -clear -alert.
  Impact: Member 0 will not be able to service requests until
  this alert has been cleared and the member returns to its home host.
  ```
- Check the `<instance_owner>.nfy log` for information about when the failure occurred. Look in this member's db2diag log file for more details on why the failure occurred. Look for error messages that are related to `db2rstar` or `db2rstop` in the db2diag log file.
- The system error log for the affected host can also be consulted if the cause of the error is still unknown. For example:
  - In the output shown previously, member 0 is not started.
  - Login to host01 and view the system error log by running the **errpt -a** command (AIX) or looking at the `/var/log/messages` (Linux).
  - In the system error log, look for related log entries at the time of the failure.
- If an alert was shown from **db2cluster -list -alert**, run **db2cluster -clear -alert** after the problem is resolved, and try the **db2start** command again.

CF server failure:

Use the information in this topic to help you diagnose if a cluster caching facility (CF) component failed.

Symptoms

A Db2 instance fails to start on the execution of the **db2start** command.

Diagnosing a CF server failure

- Refer to the SQLCODEs in the **db2start** command output.
- To determine whether a CF has not started, run **db2instance -list**. This information might show CFs in a STOPPED or ERROR state if the startup has failed, depending on when the failure occurs.
  - The following example shows a sample output from **db2instance -list**

    ```
    ID      TYPE     STATE    HOME_HOST   CURRENT_HOST   ...
    --      ----     -----    ---------   ------------
    0     MEMBER   STOPPED    host01      host01
    1     MEMBER   STOPPED    host02      host02
    2     MEMBER   STOPPED    host03      host03
    128   CF       STOPPED    host04      host04
    129   CF       STOPPED    host05      host05
    ```

```
ALERT       PARTITION_NUMBER       LOGICAL_PORT    NETNAME   ...
-----       ----------------       ------------    -------
   NO                      0                  0    host01-ib0
   NO                      0                  0    host02-ib0
   NO                      0                  0    host03-ib0
   NO                      -                  0    host04-ib0
   NO                      -                  0    host05-ib0


HOSTNAME       STATE     INSTANCE_STOPPED      ALERT
--------       -----     ----------------      -----
host01        ACTIVE            NO         NO
host02        ACTIVE                   NO         NO
host03        ACTIVE                   NO         NO
host04        ACTIVE                   NO         NO
host05        ACTIVE                   NO         NO
```

- If any alerts are present, run **db2cluster -cm -list -alerts** for more information. The alerts will provide more information about what might need to be fixed (for example, a network adapter or host is offline), or point to the `cfdiag*.log` files for more information.
- Look for errors related in the CF's db2diag log file that pertain to the time when the **db2start** command was run:

```
2009-11-09-02.32.46.967563-300 I261372A332         LEVEL: Severe
PID     : 1282088              TID  : 1            KTID : 4751433
PROC    : db2start
INSTANCE: db2inst1             NODE : 000
HOSTNAME: host04
EDUID   : 1
FUNCTION: Db2, base sys utilities, sqleIssueStartStop, probe:3973
MESSAGE : Failed to start any CF.
```

- Search the sections of the db2diag log file preceding previous trace point for more information as to why the CF has not started. For example, if cluster services cannot start a CF, the db2diag log file might show:

```
2009-11-09-02.12.40.882897-300 I256778A398         LEVEL: Error
PID     : 737522               TID  : 1            KTID : 2371807if
PROC    : db2havend
INSTANCE: db2inst1             NODE : 000
EDUID   : 1
FUNCTION: Db2, high avail services, db2haOnlineResourceGroup, probe:5982
DATA #1 : <preformatted>
Timeout waiting for resource group ca_db2inst1_0-rg to be online, last known
OpState is 2
```

- Each CF writes information to the `cfdiag*.log` and dumps more diagnostic data when required. The files reside in the directory set by the database manager configuration parameter **cf_diagpath** or if not set, the **diagpath**, or `$INSTHOME/sqllib_shared/db2dump/ $m` by default.
  - CF diagnostic log files (`cfdiag-<timestamp>.<cf_id>*.log`)
    - Each of these files keeps a log of the activities that are related to a CF. Events, errors, warnings, or additional debugging information will be logged there. This log has a similar structure to the db2diag log file. A new log is created each time that a CF starts. The logging level is controlled by the **cf_diaglevel** database manager configuration parameter .
    - Note that there is a static CF diagnostic log name that always points to the most current diagnostic logging file for each CF and has the following format: `cfdiag.<cf_id>.log`
  - CF output dump diagnostic files `cfdump.YYYYMMDDhhmmssuuuuuu.<host>.<cf_id>.out`

    - These files contain information regarding CF startup and stop. There might be some additional output in these files.
  - Management LightWeight Daemon diagnostic log file (`mgmnt_lwd_log.<cf_pid>`)
    - This log file displays the log entries that pertain to the LightWeight Daemon (LWD) process for a particular CF. Errors in this log file indicate that the LWD has not started properly.
  - CF stack files (`CAPD.<cf_pid>.<tid>.thrstk`)
    - These are stack files produced by the CF when it encounters a signal. These files are important for diagnosing a problem with the CF.
  - CF trace files (`CAPD.tracelog.<cf_pid>`)
    - A default lightweight trace is enabled for the CF.
    - These trace files appear whenever the CF terminates or stops.
    - The trace files might indicate a problem with the CF, but these files are useful for diagnosing errors only when used in combination with other diagnostic data.
- If the CF process starts successfully, a startup and initialized message is written to the CF dump files.
- For example, the contents of `cfdump.20091109015035000037.host04.128.out` include a message that shows a successful process start:

```
CA Server IPC component Initialised: LWD BG buffer count: 16
              Session ID: 1d
CA Server IPC component Acknowledged LWD Startup Message
          Waiting for LWD to Configure Server
Processors: (4:4) PowerPC_POWER5 running at 1498 MHz

Cluster Accelerator initialized

Cluster Accelerator Object Information:
   OS: AIX 64-bit
   Compiler: xlC VRM (900)
   SVN Revision: 7584
   Built on: Oct 12 2009 at 17:00:54
   Executable generated with symbols
   Model Components Loaded: CACHE  LIST  LOCK
   Transport: uDAPL
   Number of HCAs: 1
   Device[0]: hca0
   CA Port[0]: 50638
   Mgmnt Port Type: TCP/IP
   Mgmnt Port: 50642
   IPC Key: 0xe50003d
   Total Workers: 4
   Conn/Worker: 128
   Notify conns: 256
   Processor Speed: 1498.0000 MHz
```

- If the `cfdump.out.*` file does not contain the "cluster caching facility initialized" line or "cluster caching facility Object Information" and other lines shown in the following example, the CF process did not start successfully. An error message might be shown instead. Contact IBM Support for more information.
- In this example, `cfdiag-20091109015035000037.128.log` contains a successful process start. If the CF did not start properly, this log might be empty or contain error messages.

```
2009-11-09-01.50.37.0051837000-300 E123456789A779 LEVEL : Event
PID       : 688182 TID :          1
HOSTNAME  : host04
FUNCTION  : CA svr_init, mgmnt_castart
```

```
MESSAGE   : CA server log has been started.
DATA #1   :
Log Level: Error
Debugging : active
Cluster Accelerator Object Information
    AIX 64-bit
    Compiler: xlC VRM (900)
    SVN Revision: 7584
    Built on Oct 12 2009 at 17:00:59
    Executable generated with symbols.
    Executable generated with asserts.
    Model Components Loaded: CACHE, LIST, LOCK
    Transport: uDAPL
    Number of HCAs: 1
    Device[0]: hca0
    CA Port[0]: 50638
    Total Workers: 4
    Conn/Worker: 128
    Notify conns: 256
    Processor Speed: 1498.000000 Mhz.
    Allocatable Structure memory: 170 MB
```

- Look for core files or stack traceback files in the CF_DIAGPATH directory.
- However, if the CFs did not started successfully, there might be a related error in the db2diag log file. For example:

```
2009-11-09-02.32.46.967563-300 I261372A332          LEVEL: Severe
PID     : 1282088              TID  : 1             KTID : 4751433
PROC    : db2start
INSTANCE: db2inst1             NODE : 000
HOSTNAME: host04
EDUID   : 1
FUNCTION: Db2, base sys utilities, sqleIssueStartStop, probe:3973
MESSAGE : Failed to start any CA.
```

- Search around the time of this message to find related errors. For example, if cluster services cannot start a CF, the db2diag log file might show:

```
2009-11-09-02.12.40.882897-300 I256778A398   LEVEL: Error
PID     : 737522               TID  : 1             KTID : 2371807
PROC    : db2havend
INSTANCE: db2inst1             NODE : 000
EDUID   : 1
FUNCTION: Db2, high avail services, db2haOnlineResourceGroup, probe:5982
DATA #1 : <preformatted>
Timeout waiting for resource group ca_db2inst1_0-rg to be online, last known
OpState is 2
```

- Look for core files or stack traceback files in the `CF_DIAGPATH` directory.
- The system error log for the affected host might also be consulted if the cause of the error is still unknown. Log onto the CF host that has not been started and view the system error log by running the **`errpt -a`** command (on Linux, look in the `/var/log/messages` file). Look for related log entries at the time of the failure. In the example shown here, login to host04 and host05, because CF 128 and CF 129 reside on these hosts.
- If an alert was shown from **`db2cluster -list -alert`**, run **`db2cluster -clear -alert`** after the problem is resolved, and then reissue the **`db2start`** command.

Identifying uDAPL over InfiniBand communication errors:

Using diagnostic logs, operating system commands and system traces you can identify and resolve uDAPL communication errors.

After you enter **db2start**, first connection activation of the database or member restart, errors might occur, as shown in the following examples of messages in a **db2diag** log file:

```
2009-04-27-15.41.03.299437-240 I9450505A370       LEVEL: Severe
PID     : 651462               TID  : 258         KTID : 2674775 PROC : db2sysc 0
INSTANCE: db2inst1             NODE : 000
EDUID   : 258                  EDUNAME: db2sysc 0
FUNCTION: Db2, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
ca_svr_connect: dat_evd_wait failed: 0xf0000

2009-04-27-15.41.03.363542-240 I9450876A367       LEVEL: Severe
PID     : 651462               TID  : 258         KTID : 2674775 PROC : db2sysc 0
INSTANCE: db2inst1             NODE : 000
EDUID   : 258                  EDUNAME: db2sysc 0
FUNCTION: Db2, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
CAConnect: cmd_connect failed: 0x80090001

2009-04-27-15.41.03.421934-240 I9451244A1356      LEVEL: Severe
PID     : 651462               TID  : 258         KTID : 2674775 PROC : db2sysc 0
INSTANCE: db2inst1             NODE : 000
EDUID   : 258                  EDUNAME: db2sysc 0
FUNCTION: Db2, Shared Data Structure Abstraction Layer ,
          SQLE_CA_CONN_ENTRY_DATA::sqleCaCeConnect, probe:622
MESSAGE : CA RC= 2148073473
DATA #1 : String, 17 bytes
CAConnect failed.
DATA #2 : CAToken_t, PD_TYPE_SD_CATOKEN, 8 bytes
0x07000000003EE0B8 : 0000 0001 1064 AF90                    .....d..
DATA #3 : CA Retry Position, PD_TYPE_SAL_CA_RETRY, 8 bytes
0
CALLSTCK:
  [0] 0x0900000012FF274C sqleCaCeConnect__23SQLE_CA_CONN_ENTRY_DATAFCP7CATokenCl
      + 0x40C
  [1] 0x0900000012FF2CF8 sqleSingleCaCreateAndAddNewConnectionsToPool__
      21SQLE_SINGLE_CA_HANDLEFCUlT1Cb + 0x278
  [2] 0x0900000012FF9188 sqleSingleCaInitialize__21SQLE_SINGLE_CA_HANDLEFRC27SQLE_
      CA_CONN_POOL_NODE_INFOCUlP13SQLO_MEM_POOL + 0x448
  [3] 0x0900000013001C50 sqleCaCpAddCa__17SQLE_CA_CONN_POOLFsCPUl + 0x350
  [4] 0x00000001000118AC sqleInitSysCtlr__FPiT1 + 0x140C
  [5] 0x0000000100013008 sqleSysCtlr__Fv + 0x4A8
  [6] 0x0900000012E15C78 sqloSystemControllerMain__FCUiPFv_iPFi_vPPvCPi + 0xD58
  [7] 0x0900000012E177AC sqloRunInstance + 0x20C
  [8] 0x0000000100006ECC DB2main + 0xAEC
  [9] 0x0900000012C99048 sqloEDUMainEntry__FPcUi + 0xA8
```

The **db2diag** log file might also show messages similar to the following ones:

```
2009-04-27-15.41.04.595936-240 I9453087A387       LEVEL: Severe
PID     : 1249362              TID  : 258         KTID : 4395063 PROC : db2sysc 1
INSTANCE: db2inst1             NODE : 001
EDUID   : 258                  EDUNAME: db2sysc 1
FUNCTION: Db2, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
xport_send: dat_ep_post_rdma_write of the MCB failed: 0x70000

2009-04-27-15.42.04.329724-240 I9505628A1358      LEVEL: Severe
PID     : 1249362              TID  : 258         KTID : 4395063 PROC : db2sysc 1
INSTANCE: db2inst1             NODE : 001
EDUID   : 258                  EDUNAME: db2sysc 1
FUNCTION: Db2, Shared Data Structure Abstraction Layer ,
          SQLE_CA_CONN_ENTRY_DATA::sqleCaCeConnect, probe:622
MESSAGE : CA RC= 2148073485
DATA #1 : String, 17 bytes
CAConnect failed.
```

```
DATA #2 : CAToken_t, PD_TYPE_SD_CATOKEN, 8 bytes
0x07000000003EE0B8 : 0000 0001 1064 AFD0                        .....d..
DATA #3 : CA Retry Position, PD_TYPE_SAL_CA_RETRY, 8 bytes
894
CALLSTCK:
  [0] 0x0900000012FF274C sqleCaCeConnect__23SQLE_CA_CONN_ENTRY_DATAFCP7CATokenCl
      + 0x40C
  [1] 0x0900000012FF2CF8 sqleSingleCaCreateAndAddNewConnectionsToPool__
      21SQLE_SINGLE_CA_HANDLEFCUlT1Cb + 0x278
  [2] 0x0900000012FF9188 sqleSingleCaInitialize__21SQLE_SINGLE_CA_HANDLEFRC27SQLE_
      CA_CONN_POOL_NODE_INFOCUlP13SQLO_MEM_POOL + 0x448
  [3] 0x0900000013001C50 sqleCaCpAddCa__17SQLE_CA_CONN_POOLFsCPUl + 0x350
  [4] 0x00000001000118AC sqleInitSysCtlr__FPiT1 + 0x140C
  [5] 0x0000000100013008 sqleSysCtlr__Fv + 0x4A8
  [6] 0x0900000012E15C78 sqloSystemControllerMain__FCUiPFv_iPFi_vPPvCPi + 0xD58
  [7] 0x0900000012E177AC sqloRunInstance + 0x20C
  [8] 0x0000000100006ECC DB2main + 0xAEC
  [9] 0x0900000012C99048 sqloEDUMainEntry__FPcUi + 0xA8
```

These messages indicate a communication error between a CF and a member. Follow these steps:

1. Locate the `pdLogCfPrintf` messages and search for the message string `CF RC=`. For example, `CF RC= 2148073491`.
2. Take the numeric value adjacent to this string; in this example it is `2148073491`. This value represents the reason code from the network or communication layer.
3. To find more details on this error, run the **db2diag** tool with the **-cfrc** parameter followed by the value. Example: `db2diag -cfrc 2148073491`.
4. If the system was recently enabled with uDAPL and InfiniBand, check your uDAPL configuration.
5. Ping the IB hostnames from each member host that is showing the previously listed errors to the CFs IB hostnames, and from the CF hosts to the IB hostnames of those members.
6. If pinging the IB hostnames fails, verify that the port state is up. To verify if the port state is up, run **ibstat -v**. In the following example, the link should be good because Physical Port Physical State has a value of Link Up, Logical Port State has a value of Active, and Physical Port State has a value of Active:

```
$ ibstat -v
------------------------------------------------------------------------------
 IB NODE INFORMATION (iba0)
------------------------------------------------------------------------------
Number of Ports:                        2
Globally Unique ID (GUID):              00.02.55.00.02.38.59.00
Maximum Number of Queue Pairs:          16367
Maximum Outstanding Work Requests:      32768
Maximum Scatter Gather per WQE:         252
Maximum Number of Completion Queues:    16380
Maximum Multicast Groups:               32
Maximum Memory Regions:                 61382
Maximum Memory Windows:                 61382
Hw Version info:                        0x1000002
Number of Reliable Datagram Domains:    0
Total QPs in use:                       3
Total CQs in use:                       4
Total EQs in use:                       1
Total Memory Regions in use:            7
Total MultiCast Groups in use:          2
Total QPs in MCast Groups in use:       2
EQ Event Bus ID:                        0x90000300
EQ Event ISN:                           0x1004
NEQ Event Bus ID:                       0x90000300
NEQ Event ISN:                        0x90101

---------------------------------------------------------------------------
 IB PORT 1 INFORMATION (iba0)
---------------------------------------------------------------------------
Global ID Prefix:                     fe.80.00.00.00.00.00.00
Local ID (LID):                       000e
Local Mask Control (LMC):             0000
Logical Port State:                   Active
Physical Port State:                  Active
Physical Port Physical State:         Link Up
Physical Port Speed:                  2.5G
Physical Port Width:                  4X
Maximum Transmission Unit Capacity:   2048
Current Number of Partition Keys:     1
Partition Key List:
  P_Key[0]:                           ffff
Current Number of GUID's:             1
Globally Unique ID List:
  GUID[0]:                            00.02.55.00.02.38.59.00
```

7. Check the Galaxy InfiniBand adapter card, InfiniBand switch, and cable connections for failures on the physical server.
8. The AIX system error log might also show related messages. You can check the error log by running the **errpt -a** command.
9. Ensure that the InfiniBand network interface, the host channel adapter, and the icm values all are Available, as shown in the following example:

   ```
   $ lsdev -C | grep ib
   fcnet0      Defined   00-08-01 Fibre Channel Network Protocol Device
   fcnet1      Defined   00-09-01 Fibre Channel Network Protocol Device
   ib0         Available          IP over Infiniband Network Interface
   iba0        Available          InfiniBand host channel adapter
   icm         Available          Infiniband Communication Manager
   ```

   - If setup was performed correctly, and the hardware is functioning correctly, all three values should be 'Available'.
   - If the network interface is not 'Available', you can change the device state manually. To change the device state manually you can use the following command:

     ```
     $ chdev -l ib0 -a state=up
     ib0 changed
     ```

   - If iba0 or icm are not in the Available state, check for errors on the device. To check for errors on the device, run **/usr/sbin/cfgmgr -vl iba0** or **/usr/sbin/cfgmgr -vl icm** as a root user.
   - If iba0 is not found or remains in the Defined state, confirm that the Host Channel Adapter is currently assigned to the host on the HMC.
10. Verify that the cf-server processes were running on the CF server hosts at the time of the error. If the CF hosts were not up, not initialized, or were restarted at that time (when performing **db2instance -list** at the time, the primary CF was PRIMARY and the secondary was in PEER), check `cfdump.out*`, `cfdiag*.log`, and core files for more details. However, if the CF servers were up and initialized at the time of the error, then there might be a uDAPL communication problem.
11. If a **db2start** command or a CONNECT statement was issued, to determine whether there is a different failure that caused these errors to appear as a side effect, see "CF server failure" on page 198.
12. If this is not the case, a trace of the failing scenario is often useful to determine the cause for the error. If CF trace was enabled, dump it. To dump

CF trace, run the following command: `db2trc cf dump` *`fileName`* where you define the value for the *fileName* parameter.

13. To enable CF trace if it was not already enabled, run the following command: `db2trc cf on -m "*.CF.xport_udapl.*.*"` .
14. IBM Service might additionally request an AIX system trace and AIX memory traces to facilitate problem determination.
15. If CF trace on `xport_udapl` and any AIX system trace were recorded, collect this information. Run the **db2support** command to collect further diagnostic logs. Run **snap -Y** as root on all hosts, and contact IBM Service for further help.

Running a trace for uDAPL over InfiniBand connections:

When reporting a problem to IBM Service, a detailed trace of the uDAPL communication layer and lower communication layers might be requested.

These instructions can be used as a general reference for taking AIX traces.

About this task

Enabling the AIX system trace facility (OFF by default) might impact your system's performance depending on how it is used. As a result, use the trace facility to diagnose connectivity problems only when directed by IBM Service. The optimal values that are to be used for the trace mode, trace buffer size, file size, and trace hooks vary depending on the type of problem. IBM Service analyzes the problem and gives recommended values for these variables.

Procedure

To trace uDAPL over InfiniBand connections:

1. Adjust any component trace levels to refine which events should be traced for certain trace hooks. For example, to include detail level events for the InfiniBand communication layer in any subsequent AIX system trace, run the following command:
   ```
   ctctrl -t network_ib -r systracedetail
   ```
2. Turn the trace on. To turn on the trace, use the **trace** command:
   ```
   trace -T buffersize -L filesize -a -j trace_hooks
   ```
   where:
   - The default trace mode is used. Two global buffers are used to continuously gather trace data, with one buffer being written to the log file, while data is gathered in the other buffer
   - **-T** *buffersize* specifies the trace buffer size
   - **-L** *filesize* specifies the output trace log file size
   - **-a** specifies to run the trace in the background
   - **-j** *trace_hooks* specifies the type of event to trace. Multiple events can be specified as a comma-separated list

   Note: Values for trace mode, buffer scope, *buffersize*, *filesize*, and *trace_hooks* depend on the problem that is being experienced. Contact IBM Service for recommended values.
3. Reproduce the problem
4. Turn off the trace. Use the **trcstop** command.

5. Dump the trace buffer to a file. Use the **trcrpt** command:

   ```
   trcrpt > filename
   ```

   where *filename* specifies the file you dump the trace buffer to. For information about the AIX tracing facility, seeAIX Information Center: Trace Facility

   AIX also supports in-memory only traces for some components. These include default on traces, that were selected such that there is minimal performance impact. The traces are not written to a disk log file without explicit action by IBM Service personnel. To increase the InfiniBand in-memory tracing to a detailed level, use the following command:

   ```
   ctctrl -t network_ib -r memtracedetail
   ```

Verifying uDAPL configurations for connectivity issues on AIX:

uDAPL connectivity issues are most commonly caused by misconfiguration. You can verify uDAPL configurations to ensure that the members can communicate with the CF.

Before you begin

You must ensure that all of the required uDAPL packages are installed on all the hosts, and that all required software meets the minimum supported levels. You can check the installed packages and versions by issuing the `lslpp -l udapl.rte` command.

Procedure

Use the following steps to verify your uDAPL configurations:

1. Verify that the InfiniBand (IB) ports are functional on all hosts, and that the physical port states are ACTIVE. To verify, issue the `ibstat -v` command. Ensure that the Logical Port State and Physical Port State are Active. You must also ensure that the Physical Port Physical state is Link Up. In the following example, the status of port 1 of HCA `iba0` is displayed.

   ```
   -------------------------------------------------------------------------------
    IB PORT 1 INFORMATION (iba0)
   -------------------------------------------------------------------------------
   Global ID Prefix:                        fe.80.00.00.00.00.00.00
   Local ID (LID):                          0011
   Local Mask Control (LMC):                0000
   Logical Port State:                      Active
   Physical Port State:                     Active
   Physical Port Physical State:            Link Up
   Physical Port Speed:                     5.0G
   Physical Port Width:                     4X
   Maximum Transmission Unit Capacity:      2048
   Current Number of Partition Keys:        1
   Partition Key List:
     P_Key[0]:                              ffff
   Current Number of GUID's:                1
   Globally Unique ID List:
     GUID[0]:                               00.02.55.00.80.2d.dd.00
   ```

2. On the CF hosts, verify that the IP address associated with the IB ports matches the IP addresses used for the net names for the CF entry in the `db2nodes.cfg` file.

   a. View the IP address that is associated with the IB ports on the CF host. To view the IP address that is associated with the IB port, run the `ifconfig -a` command. The IP address can be found by looking at the address that is associated with the `inet` field as shown:

      ```
      coralpib23:/coralpib23/home/lpham> ifconfig -a
      ib0: flags=e3a0063<UP,BROADCAST,NOTRAILERS,RUNNING,ALLCAST,MULTICAST,LINK0,LINK1,GROUPRT,64BIT>
         inet 10.1.1.23 netmask 0xffffff00 broadcast 10.1.1.255
          tcp_sendspace 262144 tcp_recvspace 262144 rfc1323 1
      ```

      In the output, `ib0` is the interface name. The status is UP, and the IP address is 10.1.1.23. It is important to ensure that the interface status is up.

b. Ensure the network names for the CF in the `db2nodes.cfg file` match with the IP addresses for the intended IB port to use for the CF. You must also ensure that the name can be pinged, and is reachable from all hosts on the cluster.

   From each member host, run a ping command against the network names that are associated with the CF entry in the `db2nodes.cfg` file. Observe the IP address returned. The IP address must match the IP address that is associated with the IB port configuration at the CF host, as in the `ifconfig -a` output.

   Note: When you ping an IP address on a different subnet, the pings are unsuccessful. This occurs when you have multiple subnet masks for each interface when there are multiple interfaces defined for the CF. In this case, from the member, ping the target IP address on the CF host that has the same subnet mask as the interface on the member host.

3. Verify that the uDAPL interface is configured in the `/etc/dat.conf` file on all hosts, and that the right adapter port value is used. Since Db2 pureScale uses uDAPL 2.0, look for the first entry that has `u2.0` in the second column with the matching interface name and port number. The following entry might look similar to the entry in your `/etc/dat.conf` file:

```
hca2 u2.0 nonthreadsafe default /usr/lib/libdapl/libdapl2.a(shr_64.o) IBM.1.1 "/dev/iba0 1 ib0" " "
```

   In the example, `hca2` is the unique transport device name for the uDAPL interface. The `u2.0` indicates that the entry is for a uDAPL 2.0 application. You must ensure that the `/usr/lib/libdapl/libdapl2.a` file exists for it is the uDAPL shared library. The `/dev/iba0 1 ib0` output is the uDAPL provider-specific instance data. In this case, the adapter is `iba0`. The port is 1, and the interface name is `ib0`.

   If the CF is configured with multiple interfaces by using multiple netnames in the `db2nodes.cfg` file, you must ensure that all the interfaces are defined in the `dat.conf` file.

4. Ensure that the port value specified on the client connect request match the port value the CF listens on. You must ensure that the CF port values are the same in the `/etc/services` files for all hosts in the cluster.

   a. To determine the port value that is used for the CF, look in the CF diagnostic log file. In the `cfdiag_<`*`timestamp`*`>.<`*`id`*`>.log` file, look for the value that is associated with the `CA Port[0]` field as part of the prolog information at the beginning of the log file. In the following example, the port value for the CF is 37761.

```
Cluster Accelerator Object Information
    AIX 64-bit
    Compiler: xlC VRM (900)
    SVN Revision:  4232013
    Build mode: PERFORMANCE
    Built on Apr 23 2013 at 12:52:24
    Executable generated with symbols.
    Executable generated without asserts.
    Model Components Loaded: CACHE, LIST, LOCK
    Transport: uDAPL
    Number of HCAs: 1
    Device[0]: hca2
    CA Port[0]: 37761
    Total Workers: 1
    Conn/Worker: 128
    Notify conns: 1024
    Processor Speed: 5000.000000 Mhz.
    Allocatable Structure memory: 494 MB
```

   b. To determine the port value that is used by the member on the connect request, look for the PsOpen event in the Db2 member diagnostic log (`db2diag.log`) file. Look for the value of the `caport` field. In the following example, the port value for the target CF is also 37761.

```
2013-04-29-16.00.56.371442-240 I80540A583           LEVEL: Event
PID     : 10354874             TID : 772            PROC : db2sysc 0
INSTANCE: lpham                NODE : 000
HOSTNAME: coralpib23
EDUID   : 772                  EDUNAME: db2castructevent 0
```

```
FUNCTION: Db2, Shared Data Structure Abstraction Layer for CF, SQLE_SINGLE_CA_HANDLE::sqleSingleCfOpenAndConnect, probe:1264
DATA #1 : <preformatted>
PsOpen SUCCESS: hostname:coralpib23-ib0 (member#: 128, cfIndex: 1) ; device:hca2 ; caport:37761 ; transport: UDAPL
Connection pool target size = 9 conn (seq #: 3 node #: 1)
```

Verifying uDAPL configurations for connectivity issues on Linux:

uDAPL connectivity issues are most commonly caused by misconfiguration. You can verify uDAPL configurations to ensure that the members can communicate with the CF.

Before you begin

You must ensure that all of the required OFED packages are installed on all the hosts, and that all required software meets the minimum supported levels. You can check the installed packages and versions by issuing the `rpm -qa | grep ofed` command.

Procedure

Use the following steps to verify your uDAPL configurations:

1. Examine the physical port states by running the `ibstat -v` command. Ensure that the State is Active, and the Physical State is LinkUp as shown in the following example:

   ```
   CA 'mthca0'
           CA type: MT25208 (MT23108 compat mode)
           Number of ports: 2
           Firmware version: 4.7.400
           Hardware version: a0
           Node GUID: 0x0005ad00000c03d0
           System image GUID: 0x0005ad00000c03d3
           Port 1:
                   State: Active
                   Physical state: LinkUp
                   Rate: 10
                   Base lid: 16
                   LMC: 0
                   SM lid: 2
                   Capability mask: 0x02510a68
                   Port GUID: 0x0005ad00000c03d1
           Port 2:
                   State: Down
                   Physical state: Polling
                   Rate: 10
                   Base lid: 0
                   LMC: 0
                   SM lid: 0
                   Capability mask: 0x02510a68
                   Port GUID: 0x0005ad00000c03d2
   ```

   If the port State is not Active, check the cable for connectivity.
2. On the CF hosts, verify that the IP address associated with the IB ports matches the IP addresses used for the net names for the CF entry in the `db2nodes.cfg` file.
   a. View the IP address that is associated with the IB ports on the CF host. To view the IP address that is associated with the IB port, run the `ifconfig -a` command. The IP address can be found by looking at the address that is associated with the `inet addr` field as shown:

      ```
      coralxib20:/home/svtdbm3 >ifconfig -a
      ib0       Link encap:UNSPEC  HWaddr 80-00-04-04-FE-80-00-00-00-00-00-00-00-00-00-00
                inet addr:10.1.1.120  Bcast:10.1.1.255  Mask:255.255.255.0
                inet6 addr: fe80::205:ad00:c:3d1/64 Scope:Link
                UP BROADCAST RUNNING MULTICAST  MTU:65520  Metric:1
      ```

```
RX packets:18672 errors:0 dropped:0 overruns:0 frame:0
TX packets:544 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:256
RX bytes:2198980 (2.0 Mb)  TX bytes:76566 (74.7 Kb)
```

In the output, `ib0` is the interface name. The status is UP, and the IP address is 10.1.1.120. It is important to ensure that the interface status is up.

b. Ensure the network names for the CF in the `db2nodes.cfg file` match with the IP addresses for the intended IB port to use for the CF. You must also ensure that the name can be pinged, and is reachable from all hosts on the cluster.

From each member host, run a ping command against the network names that are associated with the CF entry in the `db2nodes.cfg` file. Observe the IP address returned. The IP address must match the IP address that is associated with the IB port configuration at the CF host, as in the `ifconfig -a` output.

Note: When you ping an IP address on a different subnet, the pings are unsuccessful. This occurs when you have multiple subnet masks for each interface when there are multiple interfaces defined for the CF. In this case, from the member, ping the target IP address on the CF host that has the same subnet mask as the interface on the member host.

3. Verify that the uDAPL interface is configured in the `/etc/dat.conf` file on all hosts, and that the right adapter port value is used. Since Db2 pureScale uses uDAPL 2.0, look for the first entry that has `u2.0` in the second column with the matching interface name and port number. On Linux, the adapter port value is not used, and is "0". The following entry might look similar to the entry in your `/etc/dat.conf` on SLES, or `/etc/rdma/dat.conf` on RHEL file:

```
ofa-v2-ib0 u2.0 nonthreadsafe default libdaplofa.so.2 dapl.2.0 "ib0 0" ""
```

In the output, `ofa-v2-ib0` is the unique transport device name for the uDAPL interface. The `u2.0` indicates that the entry is for a uDAPL 2.0 application. You must ensure that the `libdaplofa.so.2` file exists for it is the uDAPL shared library. The `ib0 0` output is the uDAPL provider-specific instance data. In this case, the adapter is `ib0`, and the port is "0", since it is not used.

If the CF is configured with multiple interfaces by using multiple netnames in the `db2nodes.cfg` file, you must ensure that all the interfaces are defined in the `dat.conf` file.

4. Ensure that the port value specified on the client connect request match the port value the CF listens on. You must ensure that the CF port values are the same in the `/etc/services` files for all hosts in the cluster.

a. To determine the port value that is used for the CF, look in the CF diagnostic log file. In the `cfdiag_`*`<timestamp>`*`.`*`<id>`*`.log` file, look for the value that is associated with the `CA Port[0]` field as part of the prolog information at the beginning of the log file.

b. To determine the port value that is used by the member on the connect request, look for the PsOpen event in the Db2 member diagnostic log (`db2diag.log`) file. Look for the value of the `caport` field.

Obtaining a file system trace:

When reporting a problem to IBM Service, you might be asked for a detailed trace for the Db2 cluster file system, which is based on IBM Spectrum Scale. Use the following information to take a IBM Spectrum Scale trace.

About this task

Enabling the trace facility (OFF by default) might impact the performance of your system. As a result, use the trace facility only when directed by IBM Technical Support. The optimal values that are to be used for the trace buffer size, file size, and tracing levels vary depending on the type of problem. IBM Technical Support analyzes the problem and gives recommended values for these variables.

The **mmchconfig** command can be used to configure IBM Spectrum Scale tracing parameters.

The configuration changes are automatically applied to all nodes in the cluster.

Procedure

To obtain a IBM Spectrum Scale trace, run the following steps as the root user:

1. Make sure the `/tmp/mmfs` directory exists on all nodes. Trace reports and internal dumps are written to this directory.
2. Set up the trace level on each IBM Spectrum Scale cluster by entering the following command:
   ```
   mmchconfig trace = trace_hooks
   ```
   where *trace_hooks* specifies the type of event to trace.

   Note: Values for the trace hooks depend on the problem that is being experienced. Contact IBM Technical support for recommended values.
3. Increase the size of the trace buffer and trace file by entering the following command:
   ```
   mmchconfig traceFileSize = filesize
   ```
   where *filesize* specifies the output trace log file size

   Note: The value for *filesize* depends on the problem that is being experienced. Contact IBM Technical support for recommended values.
4. To start the trace run the **mmtrace** command on all hosts that are affected by the problem.
5. To check whether the trace is started run **ps -ef | grep trace**. The command `/bin/trace` might be displayed.
6. Reproduce the problem that you want to trace.
7. Dump the trace buffer by entering the **mmtrace** command on each IBM Spectrum Scale node.
8. Turn off the trace by entering the following command:
   ```
   mmtrace stop
   ```

Frequently asked questions about post-installation problems:

Use the answers in this list of frequently asked questions to help you resolve problems that arise after the installation of the IBM Db2 pureScale Feature.

What if my members or cluster caching facilities are not in STARTED, PEER, or CATCHUP state?

If one of your members or cluster caching facilities is not in the STARTED, PEER, or CATCHUP state, perform the following steps:

1. Stop the instance by issuing the **db2stop** command.
2. Restart the instance by issuing the **db2start** command.
3. Review the messages from the failed **db2start** command, and resolve the problem.
4. If the problem persists, review the db2diag log file, which might have more details about the problem.
5. If reviewing the file does not yield a probable cause of the problem, perform the following steps:
   a. Drop the member or cluster caching facility by issuing the **db2iupdt -drop** command.
   b. Add the member or cluster caching facility again by issuing the **db2iupdt -add** command.

What if the db2start command fails with SQL1517N?

This error indicates that there are inconsistencies between the `db2nodes.cfg` file and the cluster manager. See the "Repairing the cluster manager resource model" topic for more details on how to repair the cluster manager resource model.

What if I can't create my database?

If you attempted to create a database using the create database command and received an SQL1032N error (No START DATABASE MANAGER command was issued). Ensure that you are logged on to a host that is acting as a Db2 member. You cannot create a database on a host that is acting as a cluster caching facility. Log onto a Db2 member and issue the CREATE DATABASE command again.

What if the db2iupdt -drop command failed to shrink the RSCT peer domain?

To manually remove failed hosts from the RSCT peer domain:
1. Check whether there are still resources attached to the peer domain by logging on to the failed host as the root user and issuing the **lssam** command.[3] The **lssam** command shows all resources in the domain, not just those on a particular host. Review the listing to ensure that the host that you are attempting to drop does not have any resources attached.
2. If no resources are attached, from the installation-initiating host (IIH), enter `db2cluster -cm -remove -host` *`host_name`* .
3. If there are still resources attached, perform the following steps:
   a. From the IIH, switch to the instance owner by entering `su -` *`instance_owner`* .
   b. Remove the resource by entering `db2cluster -cm -delete -resources`. This step deletes all resources in the cluster.
   c. Switch back to root.
   d. Remove the failed hosts from the RSCT peer domain by entering `db2cluster -cm -remove -host` *`host_name`*.
   e. Switch to the instance owner again.
   f. Re-create the resources on the hosts that did not fail by entering `db2cluster -cm -create -resources`.
   g. Switch back to root.

3. For more information on Tivoli SA MP commands, see this https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Documentation%20Central/page/Tivoli%20System%20Automation%20for%20Multiplatforms.

Host or member issues

When you encounter a problem, it is important to diagnose and then troubleshoot the problem. Details are given on how to identify alerts or errors and to diagnose and troubleshoot symptoms that might occur in a Db2 pureScale environment.

In a Db2 pureScale environment, the **db2instance -list** command can be run from any host where a member or cluster caching facility is located to get information about problem symptoms that might occur.

The three distinct state elements member,cluster caching facility and host can show symptoms such as alerts, errors and WAITING_FOR_FAILBACK.

If the database is unavailable, reporting of some data is still possible if the Db2 pureScale instance manager is still available on the host because you can use the **db2instance -list** command if there is no current database connection or if the instance is stopped.

Further basic diagnostic steps include:

1. Running the **db2cluster (-cm) -list -alert** command to check for alerts
2. Checking the `db2diag` log file for data diagnostic information and error messages
3. Checking the `DIAGPATH` directory for any diagnostic files that are produced and `FODC` directories
4. Checking the `CF_DIAGPATH` directory for any cluster caching facility diagnostic data produced
5. Running the **errpt -a** command to view the system error log. Look for related log entries at the time of the failure.

You can monitor the state of hosts members and cluster caching facilities.

Summary of instance status information from the db2instance -list command:

This topic provides a summary overview of the states returned by the **db2instance -list** command, which you can use to look up troubleshooting information.

Use the following table to obtain more information about output from the **db2instance -list** command.

Table 10. Permutations based on the output of the`db2instance -list` *command*

Member State	Member Alert	Host State	Host Alert	Further information
WAITING_ FOR_ FAILBACK	YES	INACTIVE	YES	See "Member WAITING_FOR_FAILBACK, alert, corresponding host is in INACTIVE state with alert" on page 213
WAITING_ FOR_ FAILBACK	NO	INACTIVE	YES	See "Member WAITING_FOR_FAILBACK, no alerts, corresponding host is in INACTIVE state with alert" on page 219
WAITING_ FOR_ FAILBACK	NO	ACTIVE	YES	See "Member WAITING_FOR FAILBACK, no alerts, the corresponding host is in ACTIVE state with an alert" on page 220
STARTED	YES	ACTIVE	NO	See "Member shows alert while in STARTED state, all hosts are in ACTIVE state with no alerts" on page 214

Table 10. Permutations based on the output of the **db2instance -list** *command (continued)*

Member State	Member Alert	Host State	Host Alert	Further information
STARTED	NO	ACTIVE	NO	The output of the "db2instance -list" command shows the primary cluster caching facility switched state from PRIMARY to PEER, and the secondary cluster caching facility switched state from PEER to PRIMARY. See “Primary role moves between cluster caching facility hosts” on page 230
ERROR	YES	ACTIVE	NO	See “Member is in ERROR state” on page 218

Member or host with alert:

If the output of the **db2instance -list** command shows that a member or a host has an alert, use the following topics that match the alerts on your instance to help diagnose the problem.

Member WAITING_FOR_FAILBACK, alert, corresponding host is in INACTIVE state with alert:

The output of the **db2instance -list** command shows that at least one member is in the WAITING_FOR_FAILBACK state with an alert and that one or more corresponding hosts are in the INACTIVE state with alerts.

This is a sample output from the **db2instance -list** command where there are three members and two cluster caching facilities

```
ID        TYPE          STATE                 HOME_HOST   CURRENT_HOST    ALERT   PARTITION_NUMBER        LOGICAL_PORT     NETNAME
--        ----          -----                 ---------   ------------    -----   ----------------        ------------     -------
0         MEMBER        WAITING_FOR_FAILBACK  hostA       hostC           YES                    0                   1     hostC-ib0
1         MEMBER        STARTED               hostB       hostB           NO                     0                   0     hostB-ib0
2         MEMBER        STARTED               hostC       hostC           NO                     0                   0     hostC-ib0
128       CF            PRIMARY               hostD       hostD           NO                     -                   0     hostD-ib0
129       CF            PEER                  hostE       hostE           NO                     -                   0     hostE-ib0

HOSTNAME     STATE        INSTANCE_STOPPED ALERT
--------     -----        ---------------- -----
hostA        INACTIVE     NO               YES
hostB        ACTIVE       NO               NO
hostC        ACTIVE       NO               NO
hostD        ACTIVE       NO               NO
hostE        ACTIVE       NO               NO
```

Member 0 experienced a problem with its home host, hostA, and is running in light mode on hostC.

Member 0 shows that an ALERT has occurred. Run **db2cluster -cm -list -alert** to find out what the ALERT is for.

Because hostA is unavailable, the state for the host is set to INACTIVE. Member 0 can not fail back while hostA is in the INACTIVE state; member 0 remains in the WAITING_FOR_FAILBACK state. Because restart light failed on hostB, an alert was raised as an indication to the administrator to investigate the problem.

If hostA becomes available again, its state will change from INACTIVE to ACTIVE. Member 0 will fail back to hostA, and its state will change from WAITING_FOR_FAILBACK to STARTED. This is covered in further details at .

For information about diagnosing this symptom, see the link "HostA fails, restart light works on a hostC, but not the first hostB"

Member shows alert while in STARTED state, all hosts are in ACTIVE state with no alerts:

This symptom observed occurs as a follow on from the the scenario "Member WAITING_FOR_FAILBACK, alert, corresponding host(s) is in INACTIVE state with alert". The output of the **db2instance -list** command shows a member with an alert while its state is STARTED. All hosts are in ACTIVE state with no alerts.

This is a sample output from the **db2instance -list** command showing a three member, two cluster caching facility environment:

```
ID       TYPE             STATE          HOME_HOST   CURRENT_HOST    ALERT   PARTITION_NUMBER   LOGICAL_PORT    NETNAME

--       ----             -----          ---------   ------------    -----   ----------------   ------------    -------
0        MEMBER           STARTED        hostA       hostA           YES                    0              0    hostA-ib0
1        MEMBER           STARTED        hostB       hostB           NO                     0              0    hostB-ib0
2        MEMBER           STARTED        hostC       hostC           NO                     0              0    hostC-ib0
128      CF               PRIMARY        hostD       hostD           NO                     -              0    hostD-ib0
129      CF               PEER           hostE       hostE           NO                     -              0    hostE-ib0

HOSTNAME              STATE      INSTANCE_STOPPED ALERT
--------              -----      ---------------- -----
hostA                 ACTIVE     NO               NO
hostB                 ACTIVE     NO               NO
hostC                 ACTIVE     NO               NO
hostD                 ACTIVE     NO               NO
hostE                 ACTIVE     NO               NO
```

Member 0 still has its alert flagged as an indicator to the administrator to investigate the problem that occurred on hostB that prevented the member from performing a restart light.

Note: The hostB problem is detailed in the parent topic Member WAITING_FOR_FAILBACK, alert, corresponding host(s) is in INACTIVE state with alert.

For information about diagnosis, see the related link "Host (A) fails, restart light works on a host (C), but not the first host (B), failback N/A".

Restart light fails on one host, but completes on another host:

During a restart light, a member fails over to a guest host, so that the recovery process can complete. Use the information in this topic to help you diagnose why a restart light on the initial guest host is unsuccessful, but then completes successfully on a second guest host.

Symptoms

The following sample output from the **db2instance -list** command shows an environment with three members and two cluster caching facilities:

```
ID       TYPE       STATE                  HOME_HOST    CURRENT_HOST    ALERT   PARTITION_NUMBER        LOGICAL_PORT    NETNAME
--       ----       -----                  ---------    ------------    -----   ----------------        ------------    -------
0        MEMBER     WAITING_FOR_FAILBACK   hostA        hostC           YES                    0                   1    hostC-ib0
1        MEMBER     STARTED                hostB        hostB           NO                     0                   0    hostB-ib0
2        MEMBER     STARTED                hostC        hostC           NO                     0                   0    hostC-ib0
128      CF         PRIMARY                hostD        hostD           NO                     -                   0    hostD-ib0
129      CF         PEER                   hostE        hostE           NO                     -                   0    hostE-ib0

HOSTNAME    STATE      INSTANCE_STOPPED ALERT
--------    -----      ---------------- -----
hostA       INACTIVE   NO               YES
hostB       ACTIVE     NO               NO
hostC       ACTIVE     NO               NO
hostD       ACTIVE     NO               NO
hostE       ACTIVE     NO               NO
```

Member 0 experienced a problem with its home host, hostA, and attempted a restart light on hostB. However, the restart light failed on hostB. The member then attempted a restart light on hostC, which was successful.

If hostA becomes available again, its state will change from INACTIVE to ACTIVE. member 0 will fail back to hostA, and the state of the member will change from WAITING_FOR_FAILBACK to STARTED.

```
ID       TYPE            STATE          HOME_HOST   CURRENT_HOST    ALERT   PARTITION_NUMBER       LOGICAL_PORT    NETNAME
--       ----            -----          ---------   ------------    -----   ----------------       ------------    -------
0        MEMBER          STARTED        hostA       hostA           YES                    0                  0    hostA-ib0
1        MEMBER          STARTED        hostB       hostB           NO                     0                  0    hostB-ib0
2        MEMBER          STARTED        hostC       hostC           NO                     0                  0    hostC-ib0
128      CF              PRIMARY        hostD       hostD           NO                     -                  0    hostD-ib0
129      CF              PEER           hostE       hostE           NO                     -                  0    hostE-ib0

HOSTNAME             STATE      INSTANCE_STOPPED ALERT
--------             -----      ---------------- -----
hostA                ACTIVE     NO               NO
hostB                ACTIVE     NO               NO
hostC                ACTIVE     NO               NO
hostD                ACTIVE     NO               NO
hostE                ACTIVE     NO               NO
```

Troubleshooting steps

To help troubleshoot the restart light failure on hostB, take one or both of the following steps:

- Check the db2diag log file for information about the failure, and then investigate it.

  The following sample output shows the restart light attempt on hostB:

  ```
  2009-08-27-23.37.52.416270-240 I6733A457           LEVEL: Event
  PID     : 1093874             TID  : 1            KTID : 2461779
  PROC    : db2star2
  INSTANCE:                     NODE : 000
  HOSTNAME: hostB
  EDUID   : 1
  FUNCTION: Db2, base sys utilities, DB2StartMain, probe:3368
  MESSAGE : Idle process taken over by member
  DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
  996
  DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
  0
  ```

  Check the `diag` messages to analyze the errors corresponding to the restart light failure on hostB.
- See "Diagnosing a host reboot with a restart light" on page 233 for steps to diagnose the host failure on hostA.
- See "Diagnosing a cluster file system failure that occurred during restart light" for an example on how to troubleshoot this scenario.
- After you diagnose the problem, clear the alert for the member.

Diagnosing a cluster file system failure that occurred during restart light:

A member attempts to perform a restart light, but a cluster file system error occurs, which causes the restart light to fail.

Symptoms

The objective of this topic is to diagnose the cause of the failure. This is a sample output from the **`db2instance -list`** command showing a three member, two cluster caching facility environment:

```
ID       TYPE            STATE            HOME_HOST    CURRENT_HOST    ALERT   PARTITION_NUMBER   LOGICAL_PORT    NETNAME
--       ----            -----            ---------    ------------    -----   ----------------   ------------    -------
0        MEMBER          RESTARTING       hostA        hostB           No                     0              1    hostB-ib0
1        MEMBER          STARTED          hostB        hostB           NO                     0              0    hostB-ib0
2        MEMBER          STARTED          hostC        hostC           NO                     0              0    hostC-ib0
128      CF              PRIMARY          hostD        hostD           NO                     -              0    hostD-ib0
129      CF              PEER             hostE        hostE           NO                     -              0    hostE-ib0

HOSTNAME       STATE      INSTANCE_STOPPED ALERT
--------       -----      ---------------- -----
hostA          INACTIVE   NO               YES
```

```
hostB         ACTIVE     NO              NO
hostC         ACTIVE     NO              NO
hostD         ACTIVE     NO              NO
hostE         ACTIVE     NO              NO
```

There is a state of `RESTARTING` for member 0 occurring on hostB. The home host for member 0 is hostA. This output indicates member 0 is actively performing a restart light on hostB

HostA has a state of `INACTIVE` with an corresponding alert. This indicates an abnormal host shutdown such as a power failure or a failure to access the host due to a network communication failure.

A subsequent **`db2instance -list`** output shows that the restart light failed on hostB. The member then attempts the restart light on hostC and is successful. Member 0 is left in the WAITING_FOR_FAILBACK state because hostA remains offline.

```
ID        TYPE             STATE                HOME_HOST    CURRENT_HOST    ALERT   PARTITION_NUMBER    LOGICAL_PORT    NETNAME
--        ----             -----                ---------    ------------    -----   ----------------    ------------    -------
0         MEMBER           WAITING_FOR_FAILBACK hostA       hostC            YES                   0               1    hostC-ib0
1         MEMBER           STARTED              hostB        hostB           NO                     0               0    hostB-ib0
2         MEMBER           STARTED              hostC        hostC           NO                     0               0    hostC-ib0
128       CF               PRIMARY              hostD        hostD           NO                     -               0    hostD-ib0
129       CF               PEER                 hostE        hostE           NO                     -               0    hostE-ib0

HOSTNAME      STATE      INSTANCE_STOPPED ALERT
--------      -----      ---------------- -----
hostA         INACTIVE   NO               YES
hostB         ACTIVE     NO               NO
hostC         ACTIVE     NO               NO
hostD         ACTIVE     NO               NO
hostE         ACTIVE     NO               NO
```

If hostA comes online, the host state will return to ACTIVE, and member 0 will fail back to hostA as shown in the following **`db2instance -list`** output

```
ID        TYPE          STATE        HOME_HOST   CURRENT_HOST    ALERT   PARTITION_NUMBER    LOGICAL_PORT    NETNAME
--        ----          -----        ---------   ------------    -----   ----------------    ------------    -------
0         MEMBER        STARTED      hostA       hostA           YES                    0               0    hostA-ib0
1         MEMBER        STARTED      hostB       hostB           NO                     0               0    hostB-ib0
2         MEMBER        STARTED      hostC       hostC           NO                     0               0    hostC-ib0
128       CF            PRIMARY      hostD       hostD           NO                     -               0    hostD-ib0
129       CF            PEER         hostE       hostE           NO                     -               0    hostE-ib0

HOSTNAME      STATE      INSTANCE_STOPPED ALERT
--------      -----      ---------------- -----
hostA         ACTIVE     NO               NO
hostB         ACTIVE     NO               NO
hostC         ACTIVE     NO               NO
hostD         ACTIVE     NO               NO
hostE         ACTIVE     NO               NO
```

Diagnosis and resolution

To help diagnose the restart light error, take one or more of the following steps:

- Check the `$INSTHOME/sqllib/db2dump/db2diag.log` file. There should be a message corresponding to the time of the restart light attempt on hostB, as shown in the following output:

  ```
  2009-08-27-23.37.52.416270-240 I6733A457           LEVEL: Event
  PID     : 1093874              TID  : 1           KTID : 2461779
  PROC    : db2star2
  INSTANCE:                      NODE : 000
  HOSTNAME: hostB
  EDUID   : 1
  FUNCTION: Db2, base sys utilities, DB2StartMain, probe:3368
  MESSAGE : Idle process taken over by member
  DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
  996
  DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
  0
  ```

- Check the db2diag log file to analyze the error message corresponding to the restart light failure on hostB. The following output is generated during the restart light attempt:

```
2009-08-27-23.37.53.260582-240 E596165A1624       LEVEL: Error (OS)
PID     : 667736               TID  : 1           PROC : db2rocm
INSTANCE:                      NODE : 000
HOSTNAME: hostB
EDUID   : 1
FUNCTION: Db2, oper system services, sqloflock, probe:100
MESSAGE : ZRC=0x83000037=-2097151945
CALLED  : OS, -, fcntl
OSERR   : EINPROGRESS (55) "Operation now in progress"
DATA #1 : File handle, PD_TYPE_SQO_FILE_HDL, 8 bytes
  File Handle              = 3
  File System Block Size   = 32768 bytes
  File System Type         = mmfs
  File Handle Flags :
    Require Sector Align   = No
    DIO/CIO Mode           = No
    Raw Block Device       = No
    Reserved Handle        = No
    Flush On Close         = No
    Thread-Level Lock      = No
    Write-through Mode     = No
    File Not Tracked       = No
```

The output shows that the fcntl() operating system function was called and generated an error message and code of `EINPROGRESS (55) "Operation now in progress"`. The `File System Type` is `mmfs`, which is the General Parallel File System (GPFS).

- Run the **`errpt -a`** operating system command to view the contents of the `AIX errpt` system log. In the scenario, as shown in the following sample output, the `AIX errpt` log from hostB contains `MMFS_*` messages (for example, `MMFS_MOREINFO`) that were generated at approximately the same time as the aforementioned restart light message. These `MMFS*` messages indicate that the problem originated in the GPFS subsystem.

```
LABEL:          MMFS_MOREINFO
IDENTIFIER:     E7AA68A1

Date/Time:       Thu Aug 27 23:37:53 EDT 2009
Sequence Number: 356562
Machine Id:      0006DA8AD700
Node Id:         hostB
Class:           S
Type:            PERM
WPAR:            Global
Resource Name:   mmfs

Description
SOFTWARE PROGRAM ERROR
```

See the topic “Diagnosing a host reboot with a restart light” on page 233 for help in diagnosing the cause of the host failure that initiated the restart light.

Further assistance for host and restart light problems is available on the IBM Technical Support website (Db2 support), where you can get information about known problems based on symptoms or error log messages. In addition, if you want IBM Technical Support to analyze your diagnostic data, obtain a **`db2support`** package by running **`db2support`** `output directory` **`-d`** `database name` **`-s`** on each node in the cluster. To submit data to IBM Technical Support, see Submitting diagnostic information to IBM Technical Support for problem determination.

Member WAITING_FOR_FAILBACK, alert, corresponding host is in ACTIVE state without alert:

The output of the **db2instance -list** command shows that at least one member is in the WAITING_FOR_FAILBACK state with an alert and that one or more corresponding hosts are in the ACTIVE state without alerts.

This is a sample output from the **db2instance -list** command where there are three members and two cluster caching facilities:

```
ID      TYPE            STATE                   HOME_HOST           CURRENT_HOST          ALERT   PARTITION_NUMBER        LOGICAL_PORT    NETNAME
--      ----            -----                   ---------           ------------          -----   ----------------        ------------    -------
0       MEMBER          STOPPED                 hostA               hostA                 NO                     0                   0          -
1       MEMBER          WAITING_FOR_FAILBACK    hostB               hostC                 YES                    0                   1          -
2       MEMBER          STARTED                 hostC               hostC                 NO                     0                   0          -
128     CF              CATCHUP                 hostB               hostB                 NO                     -                   0          -
129     CF              PRIMARY                 hostC               hostC                 NO                     -                   0          -

HOSTNAME                 STATE              INSTANCE_STOPPED        ALERT
--------                 -----              ----------------        -----
hostC                    ACTIVE             NO                      NO
hostB                    ACTIVE             NO                      NO
hostA                    ACTIVE             NO                      NO
```

Member 1 experienced a problem with its home host, hostB, and is running in light mode on hostC.

Member 1 shows that an ALERT has occurred. Running **db2cluster -cm -list -alert** provides information about the alert, as the following output example shows:

```
1.
Alert: Db2 member '1' is currently awaiting failback to its home host 'coral18'.
The cluster manager has restarted the Db2 member in restart light mode on another
host. The home host for the member is available however automatic failback to home
has been disabled.

Action: In order to move the member back its home host, the alert must be manually
cleared with the command: 'db2cluster -cm -clear -alert'.

Impact: Db2 member '1' will not be able to service requests until this alert has
been cleared and the Db2 member returns to its home host.
```

When hostB is available, the state for the host is set to ACTIVE. Member 1 cannot failback to hostB because automatic failback is disabled; member 1 remains in the WAITING_FOR_FAILBACK state until an administrator clears the alert and manually enables automatic failback by running the following command:

```
db2cluster -cm -set -option autofailback -value on
```

The instance must be restarted for the setting of the previous command to become effective.

Member has an alert:

The output of the **db2instance -list** command shows that a member is in the ALERT state but the host has an alert.

Member is in ERROR state:

Techniques to identify and diagnose a scenario when there is a member in ERROR state in the output of the **db2instance -list** command are outlined.

This is a sample output from the **db2instance -list** command showing a three member, two cluster caching facility environment:

```
ID   TYPE     STATE         HOME_HOST CURRENT_HOST  ALERT  PARTITION_NUMBER   LOGICAL_PORT  NETNAME
--   ----     -----         --------- ------------  -----  ----------------   ------------  -------
0    MEMBER   ERROR         hostA     hostA         YES                   0              0  hostA-ib0
1    MEMBER   STARTED       hostB     hostB         NO                    0              0  hostB-ib0
2    MEMBER   STARTED       hostC     hostC         NO                    0              0  hostC-ib0
128  CF       PRIMARY       hostD     hostD         NO                    -              0  hostD-ib0
129  CF       PEER          hostE     hostE         NO                    -              0  hostE-ib0

HOSTNAME              STATE      INSTANCE_STOPPED ALERT
--------              -----      ---------------- -----
hostA                 ACTIVE     NO               NO
hostB                 ACTIVE     NO               NO
hostC                 ACTIVE     NO               NO
hostD                 ACTIVE     NO               NO
hostE                 ACTIVE     NO               NO
```

This output shows that Db2 cluster services could not restart the failed member on its home host or on any other host in the cluster. As shown in the `STATE` column, member 0 is in the ERROR state. Manual intervention is required, as indicated by the alert flagged for member 0

For information about diagnosing this symptom, see "Host failed and restart light does not work on any host, failback did not happen."

Host failed and restart light does not work on any host:

A member and a host fail. Restart light is not successful on any host; failback does not happen.

Find out why the member failed and why the restart light was unsuccessful.
- On the home host, check the db2diag log file and `errpt` files for FODC directories.
- Check for cluster caching facilities states by issuing the **`db2cluster`** command, as follows:
  ```
  db2cluster -cm -list -host -state
  db2cluster -cfs -list -host -state
  ```
- See "Diagnosing a host reboot with a restart light" on page 233

Host has an alert:

The output of the **`db2instance -list`** shows at least one host is in ALERT but nomembers are in ALERT.

Member WAITING_FOR_FAILBACK, no alerts, corresponding host is in INACTIVE state with alert:

The output of the **`db2instance -list`** command shows at least one member in WAITING_FOR_FAILBACK state with no alerts, and the corresponding host is in INACTIVE state with an alert.

This is a sample output from the **`db2instance -list`** command showing a three member, two cluster caching facility environment:

```
ID   TYPE    STATE                HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
--   ----    -----                ---------  ------------  -----  ----------------  ------------  -------
0    MEMBER  WAITING_FOR_FAILBACK hostA      hostB         NO                    0             1  hostB-ib0
1    MEMBER  STARTED              hostB      hostB         NO                    0             0  hostB-ib0
2    MEMBER  STARTED              hostC      hostC         NO                    0             0  hostC-ib0
128  CF      PRIMARY              hostD      hostD         NO                    -             0  hostD-ib0
129  CF      PEER                 hostE      hostE         NO                    -             0  hostE-ib0

HOSTNAME              STATE      INSTANCE_STOPPED ALERT
--------              -----      ---------------- -----
hostA                 INACTIVE   NO               YES
```

```
hostB               ACTIVE    NO              NO
hostC               ACTIVE    NO              NO
hostD               ACTIVE    NO              NO
hostE               ACTIVE    NO              NO
```

Member 0 has experienced a problem with its home host, hostA, and performed a successful restart light on hostB.

Running the **db2instance -list** at this point shows no alerts for the member 0 after it performed a restart light. This is indicated in the ALERT column for member 0. This means member 0 was successful on its first attempt at performing restart light on hostB.

Since hostA is unavailable, the state for the host is set to INACTIVE. member 0 can not fail back while hostA is in the INACTIVE state; member 0 will therefore remain in WAITING_FOR_FAILBACK state.

If hostA becomes available again, its state will change from INACTIVE to ACTIVE. member 0 will fail back to hostA, and its state will change from WAITING_FOR_FAILBACK to STARTED. To diagnose this symptom see Diagnosing a host reboot with a restart light.

Member WAITING_FOR FAILBACK, no alerts, the corresponding host is in ACTIVE state with an alert:

The output of the **db2instance -list** command shows at least one member in WAITING_FOR_FAILBACK state with no alerts, and the corresponding host is in ACTIVE state with an alert.

This is a sample output from the **db2instance -list** command showing a three member, two cluster caching facility environment:

```
ID   TYPE    STATE                HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
--   ----    -----                ---------  ------------  -----  ----------------  ------------  -------
0    MEMBER  WAITING_FOR_FAILBACK hostA      hostB         NO                    0             1  hostB-ib0
1    MEMBER  STARTED              hostB      hostB         NO                    0             0  hostB-ib0
2    MEMBER  STARTED              hostC      hostC         NO                    0             0  hostC-ib0
128  CF      PRIMARY              hostD      hostD         NO                    -             0  hostD-ib0
129  CF      PEER                 hostE      hostE         NO                    -             0  hostE-ib0

HOSTNAME            STATE     INSTANCE_STOPPED ALERT
--------            -----     ---------------- -----
hostA               ACTIVE    NO               YES
hostB               ACTIVE    NO               NO
hostC               ACTIVE    NO               NO
hostD               ACTIVE    NO               NO
hostE               ACTIVE    NO               NO
```

This **db2instance -list** output shows that member 0 experienced a problem with its home host, hostA, and performed a successful restart light on hostB.

Running the **db2instance -list** at this point shows no alerts on the member 0 after it performed a restart light. This is indicated in the ALERT column for the member. This means member 0 was successful on its first attempt at restart light, which was performed on hostB.

There is an alert flagged for hostA, which is an indication to the administrator to investigate a problem on hostA. Although hostA is ACTIVE, there could be an issue preventing its usage in the cluster such as a file system or network problem.

For information about diagnosis for this symptom see the related link "Host fails, restarted light successfully, failback fails or cannot restart on home host ".

A member cannot restart on the home host after a successful restart light:

After a host failure, the member fails over to a guest host in restart light mode successfully, but then cannot fail back, and you are unable to restart the member on the home host manually.

Why failback might not succeed

If the host is still down
Check for hardware failure or power loss.

If `/var` is full

Db2 will not be able to start on the current host, and will attempt to restart in restart light mode on another.

Note: Failback might not occur in this situation since `/var` would still be full. Ensure that there is plenty of space. The recommendation is for at least 3GB.

- The cluster filesystem's log files are stored in `/var/adm/ras`. Any core files relating to the cluster manager (if core dumping is enabled) will go to `/var/ct/<domain>/run/mc/*/*`. Check this path to see if core files are there
  - Old files in these paths might be cleaned up along with any old system logs.
  - Increase the disk space for the `/var` file system to have at least 3 GB free space.
- Start GPFS, run **`db2cluster -cfs -start host`** `<failedHost>`
- Run **`db2cluster -cm -list -alert`** to list the alert
- Run **`db2cluster -cm -clear -alert`** to clear the alert
- If there is still a problem, run **`db2support`** `<output directory>` **`-d`** `<database name>` **`-s`** and contact IBM Technical Support.

If the host is up, but you cannot attach or restart due to communication failure, no communication through uDAPL

Perform the following diagnosis steps to determine if a communication failure through uDAPL is the reason why the failback is not succeeding.

- Ping between the failed host and the cluster caching facility hosts.
- Run **`lsdev -C | grep ib`** to verify that the InfiniBand components are in the Available State. The state should display as available.
- Use the **`ibstat -v`** to check the InfiniBand state. Verify that the port is active and that the link is up.
- Check the `cfdump.out*`, `core files`, `cfdiag*.log`, `mgmnt_lwd_log` to see if there are failures from the cluster caching facilities not starting up, . If there are failures, run **`db2instance -list`**, this will show *primary* in something other than the **STARTED** state, and *secondary* in something other than the **PEER** state.
  - If `cfdump.out` shows no initialized or object information, then it is likely that the cluster caching facility did not start successfully.
  - If the `cfdump.out` has the information, then the cluster caching facility had started successfully at some point.
- Check the physical IB network cable connections
  - See "Member timeout and failure due to uDAPL communication failure" on page 224.

- If you cannot communicate, run **db2support** `<output directory>` **-d** `<database name>` **-s** and contact IBM Technical Support.

If the host is up, but sys log shows a fibre channel card problem/disk error/SAN cable connection problem (causing a connection to the GPFS disk to fail on the host)

See "Disk failure causing Db2 cluster file system failure" on page 225 or if GPFS does not get remounted

- Run **db2cluster -cfs -list -host -state** .
- `mount | grep mmfs` to see if any results show filesystem type=mmfs.
- Check connections, cards, disks, and restart GPFS using **db2cluster -cfs -start host** `<failedHost>`.

Why the host fails

To find out why the host fails

- Check for hardware failure or power loss.
- See "Diagnosing a host reboot with a restart light" on page 233 for steps to diagnose the host failure on hostA.

Significant disk usage or full /var file path, or SQL30108N error:

This scenario illustrates a failure of the IBM Tivoli System Automation for Multiplatforms recovery resource manager daemon, which serves as the decision engine for Tivoli SA MP and is identified as IBM.RecoveryRM in the system. If the daemon fails, symptoms include significant disk usage or a full /var file path, or an SQL30108N

There are two cases for this scenario.

Case 1: Significant disk usage (or full) '/var' file system

The following scenario shows the symptoms for this error:

- If the IBM Tivoli System Automation for Multiplatforms recovery resource manager daemon fails during runtime, Tivoli SA MP will automatically try to restart the daemon so that end users or applications are not affected and database processing will continue. There will not be any noticeable symptoms. Tivoli SA MP will write diagnostic information into `/var/ct/db2domain/log/mc/` (error logs) and `/var/ct/db2domain/run/mc/` (core dumps) and `/tmp/db2_cluster_manager_spooling` (default trace directory). Check the `var/...` directories for an accumulation of diagnostic data.

The following instructions give details on diagnosis and resolution:

- If you see a continuous accumulation of diagnostic data written into the aforementioned directories, it is safe to archive old diagnostic data to an archive location. For example
  - `mv /var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp /archivepath` where `/var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp` is a Tivoli SA MP diagnostic destination path

    Note: `/archivepath` is an arbitrary file system
- It is important to monitor the `/var/ct/db2domain/log/mc/` and `/var/ct/db2domain/run/mc/` directories on a regular basis and maintain free space of at least 3 GB for the `/var` file system.

- IBM service and development teams use trace and core files for troubleshooting. For IBM Technical Support to analyze the diagnostic data, obtain a **db2support** package by running the following command on each member in the cluster
  - `db2support output_directory -d database_name -s`
- Follow these instructions to upload data to IBM Technical Support:
  - Submitting diagnostic information to IBM Technical Support for problem determination
- The IBM Technical Support website is a good source of information, where you can identify known problems based on symptoms or error log messages
  - DB2 for Linux UNIX and Windows support.

Case 2: SQL30108N

This case occurs if restarting the daemon fails with an error, resulting in a restart light onto another host in the cluster.

The following scenario shows the symptoms for this error:
- The application returns an SQL30108N error message.
- Check for an accumulation of diagnostic data under the `/var` file system. Tivoli SA MP will write diagnostic information into `/var/ct/db2domain/log/mc/` (error logs) and `/var/ct/db2domain/run/mc/` (trace and core dumps) and `/tmp/db2_cluster_manager_spooling` (default trace directory). .

The following instructions give details on diagnosis and resolution:
- Check the db2diag log file for messages similar to the following one:

  -
    ```
    DATA #6 : String, 48 bytes
    Line # : 6884---2610-403 The resource is stale.
    ```

    or

  -
    ```
    DATA #6 : String, 142 bytes
    Line # : 9578---2610-422 Cannot execute the command on
     node coralxib38.torolab.ibm.com. The resource manager
    IBM.RecoveryRM is not available.
    ```

    If you see the previous errors, this indicates that the Tivoli SA MP recovery resource manager daemon experienced a problem. Diagnostic data will be written by Tivoli SA MP to diagnose the problem.
- If there is a continuous accumulation of diagnostic data written into the `/var/ct/db2domain/log/mc/` and `/var/ct/db2domain/run/mc/` directories, it is safe to archive old diagnostic data to an archive location. For example
  - `mv /var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp /archivepath` where `/var/ct/db2domain/run/mc/IBM.RecoveryRM/trace.6.sp` is a Tivoli SA MP diagnostic destination path

    Note: `/archivepath` is an arbitrary archive file system
- It is important to monitor the `/var/ct/db2domain/log/mc/` and `/var/ct/db2domain/run/mc/` directories on a regular basis and maintain free space of at least 3 GB for the `/var` file system.
- IBM service and development teams use trace and core files for troubleshooting. If you would like IBM Technical Support to analyze the diagnostic data, obtain a **db2support** package by running the following command on each node in the cluster
  - `db2support output_directory -d database_name -s`

- Follow these instructions to upload data to IBM Technical Support:
  - Submitting diagnostic information to IBM Technical Support for problem determination
- The IBM Technical Support website is a good source of information, where you can identify known problems based on symptoms or error log messages
  - DB2 for Linux UNIX and Windows support.

Member timeout and failure due to uDAPL communication failure:

Diagnosis and resolution of uDAPL communication timeouts causing a member to sporadically fail and restart.

Symptoms

A member sporadically experiences failures during startup or normal day-to-day OLTP processing. The member is subsequently restarted successfully on its home host, or restarted light onto another host if restarting on the home host is not possible. Informational stack traceback files, dump files, and other data that is normally dumped is written into the **diagpath** and **cf_diagpath**.

A possible symptom is that there is an increased pace of disk space usage within the db2dump file system due to the writing of such diagnostics data.

Diagnosis and resolution

To diagnose and resolve this problem carry out the following steps:

- Check the ~/sqllib_shared/db2dump/ $m to see whether the member **db2diag** log file is abnormally large or many diagnostic dump directories exist within it. On checking the **db2diag** log file, there might be messages from function pdLogCaPrintf :

```
2009-01-23-17.33.23.976179-300 I54636324503       LEVEL: Severe
PID     : 602310              TID  : 15422       PROC : db2sysc 0
INSTANCE:                     NODE : 000         DB   :
APPHDL  : 0-53                APPID:
AUTHID  :
EDUID   : 15422               EDUNAME: db2agent (     ) 0
FUNCTION: Db2, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
xport_send: Timed-out waiting for completion of dat_ep_post_rdma_write of an MCB

2009-01-23-17.33.24.473642-300 I54641364467       LEVEL: Severe
PID     : 602310              TID  : 15422       PROC : db2sysc 0
INSTANCE:                     NODE : 000         DB   :
APPHDL  : 0-53                APPID:
AUTHID  :
EDUID   : 15422               EDUNAME: db2agent (    ) 0
FUNCTION: Db2, RAS/PD component, pdLogCaPrintf, probe:876
DATA #1 : <preformatted>
ClientXport.send (CARAR:) failed: 0x80090013
```

- Check the **db2diag** log file for Restart Light messages after the aforementioned diag messages. See “Restart events that might occur in Db2 pureScale environments” on page 227 for more information about the various restart messages including the Restart Light message.

```
2009-08-27-23.37.52.416270-240 I6733A457        LEVEL: Event
PID     : 1093874             TID  : 1          KTID : 2461779
PROC    : db2star2
INSTANCE:                     NODE : 001
HOSTNAME: hostC
```

```
EDUID   : 1
FUNCTION: Db2, base sys utilities, DB2StartMain, probe:3368
MESSAGE : Idle process taken over by member
DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
1
```

- After locating the pdLogCfPrintf messages, search for the diag message string `CF RC=`. For example, `CF RC= 2148073491`
- Take the numeric value adjacent to this string, in this example it is `2148073491`. This represents the reason code from the network or communications layer.
- To find more details on this error, use the **db2diag** tool. For example, `db2diag -cfrc 2148073491`
- Ping the cluster caching facility to see if it is online. If the ping is successful, gather a **db2support** package by running `db2support` *output_directory* `-d` *database_name* `-s` on each cluster and contact IBM Technical Support.
- A uDAPL trace might be requested by IBM Service for diagnosing such problems, see "Running a trace for uDAPL over InfiniBand connections" on page 205.

Disk failure causing Db2 cluster file system failure:

A failure of the Db2 cluster file system, which is based on GPFS, on one member that causes all shared file systems (including the `db2dump`) to be unmounted for that member. GPFS stayed down.

Symptoms

Note: The `db2dump` directory will only be lost if it is on the GPFS file system. If it has been set to something else (For example, **diagpath**) then this GPFS failure will not affect it.

This is a sample output from the **db2instance -list** command showing a three member, two cluster caching facility environment, where there has been a host alert:

```
db2instance -list
ID   TYPE    STATE                HOME_HOST   CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
--   ----    -----                ---------   ------------  -----  ----------------  ------------  -------
0    MEMBER  WAITING_FOR_FAILBACK hostA       hostB         NO                    0             1  hostB-ib0
1    MEMBER  STARTED              hostB       hostB         NO                    0             0  hostB-ib0
2    MEMBER  STARTED              hostC       hostC         NO                    0             0  hostC-ib0
128  CF      PRIMARY              hostD       hostD         NO                    -             0  hostD-ib0
129  CF      PEER                 hostE       hostE         NO                    -             0  hostE-ib0

HOSTNAME              STATE      INSTANCE_STOPPED ALERT
--------              -----      ---------------- -----
hostA                 ACTIVE     NO               YES
hostB                 ACTIVE     NO               NO
hostC                 ACTIVE     NO               NO
hostD                 ACTIVE     NO               NO
hostE                 ACTIVE     NO               NO
```

Diagnosis / resolution

This scenario can be identified through GPFS error messages

- Check the alert message by running **db2cluster -cm -list -alert**, for example

```
db2cluster -cm -list -alert
The host "hostA.torolab.ibm.com" is not able to access the following
file systems:
"/db2cfs/db2inst1/sqllib/db2dump".
Check the disk connections and mount the file system.
See the Db2 Knowledge Center for more details.
```

```
This alert must be cleared manually via the command
db2cluster -clear -alert -host hostA.torolab.ibm.com
While the file system is offline, the db2 members on this
host will be in restart light mode on other systems and will
be WAITING_FOR_FAILBACK
```

- Confirm that you can access the file systems in question by using the **ls** or **cd** operating system commands.

  ```
  ls /db2cfs/db2inst1/sqllib/db2dump
  cd /db2cfs/db2inst1/sqllib/db2dump
  ```

  If the file systems are inaccessible or offline, these commands will return a message indicating that the directory does not exist or is not available.
- If the file system is inaccessible by running the **ls** or **cd** commands, confirm if the file systems are considered mounted on the problematic host.
  - Using this example scenario, on hostA run

    ```
    mount |grep sqllib
    ```

    If it is not mounted the /db2cfs/db2inst1/sqllib file system will not be shown in the result set.
  - To mount the file systems in question run the command **db2cluster -cfs -mount -filesystem** *fs_name*
- Check the **db2diag** log file at the **diagpath** location. If you require further information about the failure, look for relevant messages to ascertain the problem leading to the restart light. There might be a **db2diag** log record corresponding to the time of the restart, for example

  ```
  2009-08-27-23.37.52.416270-240 I6733A457          LEVEL: Event
  PID     : 1093874            TID  : 1           KTID : 2461779
  PROC    : db2star2
  INSTANCE:                    NODE : 000
  HOSTNAME: hostB
  EDUID   : 1
  FUNCTION: Db2, base sys utilities, DB2StartMain, probe:3368
  MESSAGE : Idle process taken over by member
  DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
  996
  DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
  0
  ```

See "Restart events that might occur in Db2 pureScale environments" on page 227 for more information about the various restart messages including the Restart Light message.

- Another source of reference for diagnosing problems is the system log. Run the operating system command **errpt -a** to view the contents of the AIX errpt system log. In this scenario example, by looking at the AIX errpt log from hostA just before the time of the aforementioned Restart Light message, you see MMFS_* messages (For example MMFS_GENERIC, MMFS_PHOENIX in errpt) with text that is similar to the following text:

  ```
  message: "GPFS: 6027-752 Lost membership in cluster hostA. Unmounting file
  systems.
  ```
- Check the disks, disk connections, and fibre channel cards. The root cause in this example scenario was faulty interconnects between the host and the SAN.
- For IBM Technical Support to analyze the diagnostic data, obtain a **db2support** package by running **db2support** output_directory **-d** database_name on each member in the cluster. Follow the instructions at 'Submitting diagnostic information to IBM Technical Support for problem determination' to upload data to IBM Technical Support:

Restart events that might occur in Db2 pureScale environments:

Information on types of restart events that might occur in Db2 pureScale. Restart Light diag message should not be confused with other types of restart messages.

The following events are types of restart events that might occur in Db2 pureScale environments.

1. **Member Local Restart**

   ```
   2009-11-08-18.56.29.767906-300 I151976017E340       LEVEL: Event
   PID : 26319                    TID : 46989525792064 KTID : 26319
   PROC : db2rocm 1
   INSTANCE: inst1                NODE : 001
   HOSTNAME: hostA
   FUNCTION: Db2, high avail services, sqlhaStartPartition, probe:1636
   DATA #1 : <preformatted>
   Successful start of member

   2009-11-08-18.56.29.769311-300 I151976358E373       LEVEL: Event
   PID : 26319                    TID : 46989525792064 KTID : 26319
   PROC : db2rocm 1
   INSTANCE: inst1                NODE : 001
   HOSTNAME: hostA
   FUNCTION: Db2, high avail services, db2rocm_main, probe:1381
   DATA #1 : String, 30 bytes
   db2rocm 1 Db2 inst1 1 START
   DATA #2 : String, 7 bytes
   SUCCESS
   ```

   Note the `Successful start of member` string from function `sqlhaStartPartition`, which indicates a Local Restart has initiated.

2. **Member Restart Light**

   ```
   2009-08-27-23.37.52.416270-240 I6733A457            LEVEL: Event
   PID     : 1093874              TID  : 1             KTID : 2461779
   PROC    : db2star2
   INSTANCE:                      NODE : 001
   HOSTNAME: hostC
   EDUID   : 1
   FUNCTION: Db2, base sys utilities, DB2StartMain, probe:3368
   MESSAGE : Idle process taken over by member
   DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
   996
   DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
   1
   ```

   Note the `Idle process taken over by member` which indicates that a recovery idle process was activated to perform a Restart Light where member 1 failed over to hostC.

3. **Member crash recovery**

   ```
   2009-11-09-13.55.08.330120-300 I338293E831           LEVEL: Info
   PID     : 24616                TID  : 47881260099904 KTID : 24731
   PROC    : db2sysc 0
   INSTANCE:                      NODE : 000          DB   :
   APPHDL  : 0-52                 APPID: *N0.DB2.091109185417
   EDUID   : 24                   EDUNAME: db2agent (     ) 0
   FUNCTION: Db2, data protection services, sqlpgint, probe:430
   DATA #1 : <preformatted>
   Crash recovery decision:
   Is this member consistent? No
   Is the database marked consistent in the GLFH? No
   Is the database restore pending? No
   Is the database rollforward pending? No
   Were the CF structures valid on startup? No
   Are we performing an offline restore? No
   Are we performing group crash recovery? No
   ```

```
Are we performing member crash recovery? Yes
Are we initializing the LFS for the group? No

..............

2009-11-09-13.55.09.002292-300 I50967394E495        LEVEL: Info
PID     : 24616                TID : 47881260099904 KTID : 24731
PROC    : db2sysc 0
INSTANCE:                      NODE : 000           DB :
APPHDL  : 0-52                 APPID: *N0.DB2.091109185417
AUTHID  :
EDUID   : 24                   EDUNAME: db2agent (     ) 0
FUNCTION: Db2, recovery manager, sqlpresr, probe:3170
DATA #1 : <preformatted>
Crash recovery completed. Next LSN is 00000000006DB3D1
```

The `Are we performing member crash recovery? Yes` string shows a members crash recovery event occurred.

4. **Group Restart**
5. **Group Crash Recovery**

```
2009-11-09-22.24.41.330120-300 I338293E831           LEVEL: Info
PID     : 10900                TID  : 46949294139712 KTID : 18929
PROC    : db2sysc 2
INSTANCE:                      NODE : 002          DB   :
APPHDL  : 2-52                 APPID: *N2.DB2.091110032438
EDUID   : 24                   EDUNAME: db2agnti (       ) 2
FUNCTION: Db2, data protection services, sqlpgint, probe:430
DATA #1 : <preformatted>
Crash recovery decision:
Is this member consistent? No
Is the database marked consistent in the GLFH? No
Is the database restore pending? No
Is the database rollforward pending? No
Were the CF structures valid on startup? No
Are we performing an offline restore? No
Are we performing group crash recovery? Yes
Are we performing member crash recovery? No
Are we initializing the LFS for the group? Yes

2009-11-09-22.24.41.540562-300 E365262E436           LEVEL: Info
PID     : 10900                TID  : 46949294139712 KTID : 18929
PROC    : db2sysc 2
INSTANCE:                      NODE : 002          DB   :
APPHDL  : 2-52                 APPID: *N2.DB2.091110032438
EDUID   : 24                   EDUNAME: db2agnti (       ) 2
FUNCTION: Db2, base sys utilities, sqledint, probe:3559
MESSAGE : Crash Recovery is needed.

2009-11-09-22.24.53.177348-300 E431539E458           LEVEL: Info
PID     : 10900                TID  : 46949294139712 KTID : 18929
PROC    : db2sysc 2
INSTANCE:                      NODE : 002          DB   :
APPHDL  : 2-52                 APPID: *N2.DB2.091110032438
EDUID   : 24                   EDUNAME: db2agnti (       ) 2
FUNCTION: Db2, recovery manager, sqlpresr, probe:210
MESSAGE : ADM1527I  Group crash recovery has been initiated.

..............

2009-11-09-22.25.14.432113-300 E552559E467           LEVEL: Info
PID     : 10900                TID  : 46949294139712 KTID : 18929
PROC    : db2sysc 2
INSTANCE:                      NODE : 002          DB   :
APPHDL  : 2-52                 APPID: *N2.DB2.091110032438
```

```
EDUID   : 24                    EDUNAME: db2agnti (       ) 2
FUNCTION: Db2, recovery manager, sqlpresr, probe:3110
MESSAGE : ADM1528I  Group crash recovery has completed successfully.
```

- The following lines from the message log indicate a Group Crash Recovery event occurred.

  ```
  Are we performing group crash recovery? Yes
  Group crash recovery has been initiated.
  Group crash recovery has completed successfully.
  ```

Member ACTIVE, but host has an alert related to /var file system:

The output of the **db2instance -list** command shows that all members are in ACTIVE state, but at least one host has an alert related to the /var file system.

This is a sample output from the **db2instance -list** command showing a three-member, two-cluster caching facility environment:

```
ID   TYPE    STATE    HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
--   ----    -----    ---------  ------------  -----  ----------------  ------------  -------
0    MEMBER  STARTED  hostA      hostA         NO                    0             0        -
1    MEMBER  STARTED  hostB      hostB         NO                    0             0        -
2    MEMBER  STARTED  hostC      hostC         NO                    0             0        -
128  CF      PRIMARY  hostD      hostD         NO                    -             0        -
129  CF      PEER     hostE      hostE         NO                    -             0        -

HOSTNAME              STATE      INSTANCE_STOPPED ALERT
--------              -----      ---------------- -----
hostA                 ACTIVE     NO               YES
hostB                 ACTIVE     NO               NO
hostC                 ACTIVE     NO               NO
hostD                 ACTIVE     NO               NO
hostE                 ACTIVE     NO               NO
```

This output shows that hostA has an alert set on it. To gather more information about the alert, use the **db2cluster** command with the -cm -list -alert option. If the /var file system free space is less than 25 MB on any host, the following alert message will appear for that host:

```
Alert: Host name hostA has 10 MB free space available on /var file system. Failure
 to write to /var due to no
space will result in Db2 cluster failure.

Action: The file system requires a minimum of 25 MB free disk space is available.
 Free up space on /var file system.
The alert will be cleared automatically when a
sufficient amount of space becomes available on the filesystem.

Impact: Db2 Cluster Services may not function properly on the specified host and
may eventually lead to a Db2 instance failure.
```

Even though this alert is only informative, it is extremely important that you monitor this situation and take action to increase the file system space on the affected host.

Another possibility is that Db2 is unable to gather information about the /var file system usage for a host. In this case, the following alert message will appear for that host:

```
Alert: There was a failure to retrieve /var file system usage on host name
db2dev07. Check the db2diag.log for messages
concerning failures on host
'hostA' for /var file system.

Action: The failure needs to be resolved to clear the alert.

Impact: Db2 cluster will fail to write information about important errors and
events. This will cause Db2 cluster failure.
```

Because it is essential that the host file system usage is monitored, you should take action as soon as possible.

Member or host without any alert:

The output for the **db2instance -list** command shows no alerts for any members or hosts, but other problems might have occurred.

Primary role moves between cluster caching facility hosts:

The primary role is not on the same host as the last time the **db2instance -list** command was run, that is, the primary role has moved to another cluster caching facility host. This host change indicates a problem with the cluster caching facility at some point in the past that might need to be investigated.

This is a sample output from the **db2instance -list** command showing a three member, two cluster caching facility environment:

```
ID       TYPE             STATE         HOME_HOST
--       ----             -----         ---------
0        MEMBER           STARTED       hostA
1        MEMBER           STARTED       hostB
2        MEMBER           STARTED       hostC
128      CF               PEER          hostD
129      CF               PRIMARY       hostE

CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
------------  -----  ----------------  ------------  -------
hostA         NO                    0             0  hostA-ib0
hostB         NO                    0             0  hostB-ib0
hostC         NO                    0             0  hostC-ib0
hostD         NO                    -             0  hostD-ib0
hostE         NO                    -             0  hostE-ib0

HOSTNAME      STATE      INSTANCE_STOPPED ALERT
--------      -----      ---------------- -----
hostA         ACTIVE     NO               NO
hostB         ACTIVE     NO               NO
hostC         ACTIVE     NO               NO
hostD         ACTIVE     NO               NO
hostE         ACTIVE     NO               NO
```

Like members, each cluster caching facility will log information into the `cfdiag*.log` and dump more diagnostic data when required. The files will reside in the directory set by the database manager configuration parameter **cf_diagpath** or if not set, the **diagpath** or $*INSTHOME*/sqllib_shared/db2dump/ $m by default.

- cluster caching facility Diagnostic Log Files (`cfdiag-`*`timestamp.cf_id`*`.log`)
  - Each of these files keep a log of the activities related to a cluster caching facility. Events, errors, warnings, or additional debugging information will be logged here. This log has a similar appearance to the **db2diag** log file. A new log is created each time a cluster caching facility starts.
  - Note that there is a single static cluster caching facility diagnostic log name that always points to the most current diagnostic logging file for each cluster caching facility and has the following format: `cfdiag.`*`cf_id`*`.log`
- cluster caching facility Output Dump Diagnostic Files (`cfdump.out.`*`cf_pid.hostname.cf_id`*)
  - These files contain information regarding cluster caching facility startup and stop. There might be some additional output shown here.
- Management LWD Diagnostic Log File (`mgmnt_lwd_log.`*`cf_pid`*)

  - This log file displays the log entries of a particular cluster caching facility's LightWeight Daemon (LWD) process. Errors presented in this log file indicate the LWD has not started properly. A successful start will not have ERROR messages in the log.
- cluster caching facility stack files (`CAPD.`*`cf_pid`*`.`*`tid`*`.thrstk`)
  - These are stack files produced by the cluster caching facility when it encounters a signal. These files are important for diagnosing a problem with the cluster caching facility.
- cluster caching facility trace files (`CAPD.tracelog.`*`cf_pid`*)
  - A default lightweight trace is enabled for the cluster caching facility. These trace files appear whenever the cluster caching facility terminates or stops. These might indicate a problem with the cluster caching facility, only in combination with other diagnostic data can these files be useful in diagnosing any errors.

A startup and initialization message will be shown in the cluster caching facility dump files. For example, the message for `cfdump.out.1548476.host04.128` contains the message that shows a successful process start:

```
CA Server IPC component Initialised: LWD BG buffer count: 16
              Session ID: 1d
CA Server IPC component Acknowledged LWD Startup Message
          Waiting for LWD to Configure Server
Processors: (4:4) PowerPC_POWER5 running at 1498 MHz

Cluster Accelerator initialized

Cluster Accelerator Object Information:
   OS: AIX 64-bit
   Compiler: xlC VRM (900)
   SVN Revision: 7584
   Built on: Oct 12 2009 at 17:00:54
   Executable generated with symbols
   Model Components Loaded: CACHE  LIST  LOCK
   Transport: uDAPL
   Number of HCAs: 1
   Device[0]: hca0
   CF Port[0]: 50638
   Mgmnt Port Type: TCP/IP
   Mgmnt Port: 50642
   IPC Key: 0xe50003d
   Total Workers: 4
   Conn/Worker: 128
   Notify conns: 256
   Processor Speed: 1498.0000 MHz
```

In this example, `cfdiag-20091109015035000037.128.log` contains a successful process start. If the cluster caching facility did not start properly, this log might be either empty or contain error messages. For example:

```
2009-11-09-01.50.37.0051837000-300 E123456789A779    LEVEL    : Event
PID       : 688182 TID :         1
HOSTNAME  : host04
FUNCTION  : CA svr_init, mgmnt_cfstart
MESSAGE   : CA server log has been started.
DATA #1   :
Log Level: Error
Debugging : active
Cluster Accelerator Object Information
    AIX 64-bit
    Compiler: xlC VRM (900)
    SVN Revision: 7584
    Built on Oct 12 2009 at 17:00:59
```

```
Executable generated with symbols.
Executable generated with asserts.
Model Components Loaded: CACHE, LIST, LOCK
Transport: uDAPL
Number of HCAs: 1
Device[0]: hca0
CF Port[0]: 50638
Total Workers: 4
Conn/Worker: 128
Notify conns: 256
Processor Speed: 1498.000000 Mhz.
Allocatable Structure memory: 170 MB
```

Look for the relevant cluster caching facility diagnostic log files by looking for the `cfdiag log` that has the same CF ID as the failed cluster caching facility. For example, if CF ID 128 failed (as it did in the previous **db2instance -list** command), use the following command:

```
$ ls cfdiag*.128.log

cfdiag.128.log -> cfdiag-20091109015035000215.128.log
cfdiag-20091110023022000037.128.log
cfdiag-20091109015035000215.128.log
```

Note that `cfdiag.128.log` always points to the most current `cfdiag log` for CF 128. Look into `cfdiag-20091109015035000037.128.log` (the previous `cfdiag log`) and the **db2diag** log file at a time corresponding to 2009-11-10-02.30.22.000215 for errors.

The system error log for the affected host can also be consulted if the cause of the error is still unknown. Log onto the unstartedcluster caching facility host and view the system error log by running the **errpt -a** command (on Linux, look in the `/var/log/messages`file). In the example shown here, log in to hostD because CF 128 experienced the failure.

Multiple member restarts on the home host:

There are no apparent problems in the output of the **db2instance -list** command, and there are no alerts flagged in the ALERT column for either the members or the hosts. There might however exist a large or growing number of FODC directories, or space usage in the **diagpath** diagnostic dump directory (`$`*INSTHOME*`/sqllib/db2dump/ $m` by default) continues to grow at an abnormal pace (based on historical behavior).

This is a sample output from the **db2instance -list** command showing a three member, two cluster caching facility environment:

```
ID   TYPE    STATE    HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
--   ----    -----    ---------  ------------  -----  ----------------  ------------  -------
0    MEMBER  STARTED  hostA      hostA         NO                    0             0  hostA-ib0
1    MEMBER  STARTED  hostB      hostB         NO                    0             0  hostB-ib0
2    MEMBER  STARTED  hostC      hostC         NO                    0             0  hostC-ib0
128  CF      PRIMARY  hostD      hostD         NO                    -             0  hostD-ib0
129  CF      PEER     hostE      hostE         NO                    -             0  hostE-ib0

HOSTNAME       STATE      INSTANCE_STOPPED ALERT
--------       -----      ---------------- -----
hostA          ACTIVE     NO               NO
hostB          ACTIVE     NO               NO
hostC          ACTIVE     NO               NO
hostD          ACTIVE     NO               NO
hostE          ACTIVE     NO               NO
```

This symptom can be caused by a member restart on the home host. Ongoing or a repetition of problems causing multiple member restarts on the home host can lead to the growing space usage of the **diagpath** diagnostic dump directory (`$`*INSTHOME*`/sqllib/db2dump/ $m` by default)

- Check the *instance_owner*.nfy log for information about when the failure occurred.
- Look for entries in this member **db2diag** log file around this timestamp for more details on why the failure occurred.

  Note: Check for error messages related to db2rstar in the **db2diag** log file.
- Look for FODC directories in the **diagpath** location (or sqllib/db2dump/ $m directory by default). If there are FODC directories see the related link to "First occurrence data capture information" for instructions on how to proceed.
- If there are no FODC directories and the cause of the error is still unknown, consult the system error log for the affected host.
  - Log in to each host
  - On AIX, run the **errpt -a** command to view the system error log.
  - On Linux, look in /var/log/messages
  - Look for evidence of a host reboot event, see "Diagnosing a host reboot with a restart light."

Diagnosing a host reboot with a restart light:

A host gets rebooted during runtime and a restart light has occurred. This identifies and gives details on how to diagnose the various situations that might have occurred.

Diagnosis

This section describes how to identify a restart light that has occurred due to a host reboot.

A message will display in the db2diag log file showing a restart light event, for example

```
2009-11-02-22.56.30.416270-240 I6733A457   LEVEL: Event
PID     : 1093874             TID  : 1    KTID : 2461779
PROC    : db2star2
INSTANCE:                     NODE : 001
HOSTNAME: hostC
EDUID   : 1
FUNCTION: Db2, base sys utilities, DB2StartMain, probe:3368
MESSAGE : Idle process taken over by member
DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
1
```

A message might display in the AIX error log showing a reboot based on the time of the db2diag log file entry shown previously. Run **errpt -a** to access the AIX error log. The following scenarios are three possible reasons for this occurrence:

- **A user-initiated host shutdown and reboot has occurred**.

  To determine whether this situation occurred, look in the AIX errpt log for an entry similar to the following one:

  ```
  LABEL:           REBOOT_ID
  IDENTIFIER:      2BFA76F6

  Date/Time:       Mon Nov  2 22:56:28 EST 2009
  Sequence Number: 11878
  Machine Id:      00057742D900
  Node Id:         coralpib17
  ```

```
Class:          S
Type:           TEMP
WPAR:           Global
Resource Name:  SYSPROC

Description
SYSTEM SHUTDOWN BY USER

Probable Causes
SYSTEM SHUTDOWN

Detail Data
USER ID
           0
0=SOFT IPL 1=HALT 2=TIME REBOOT
           0
TIME TO REBOOT (FOR TIMED REBOOT ONLY)
           0
```

In this example, a user has initiated the reboot.

- **Tivoli SA MP/RSCT initiated the reboot**.

  To determine whether this situation occurred, look in the AIX errpt log for an entry similar to the following one:

```
LABEL:          TS_CRITICAL_CLNT_ER
IDENTIFIER:     75FA8C75

Date/Time:      Tue Mar 31 10:58:49 EDT 2009
Sequence Number: 358837
Machine Id:     0006DA8AD700
Node Id:        coralp08
Class:          S
Type:           PERM
WPAR:           Global
Resource Name:  cthats

Description
Critical client blocked/exited

Probable Causes
Group Services daemon was blocked too long or exited

Failure Causes
Group Services daemon blocked: resource contention
Group Services daemon blocked: protocol problems
Group Services daemon exited: internal failure
Group Services daemon exited: critical client failure


        Recommended Actions
        Group Services daemon blocked: reduce system load
        Group Services daemon exited: diagnose Group Services

Detail Data
DETECTING MODULE
rsct,monitor.C,1.124.1.3,5520
ERROR ID
6plcyp/dyWo7/lSx/p3k37....................
REFERENCE CODE

Critical client - program name
hagsd
Failure Code
BLOCKED
Action
NODE REBOOT
```

In this example, RSCT has rebooted the host to protect critical resources in the cluster.

- **A kernel panic caused a reboot**.

  Reviewing a KERNEL_PANIC message in the AIX errpt log, or the message written before it, can help identify the underlying trigger of a kernel panic. If the LABEL, PANIC STRING, or Detail Data fields in the message contain MMFS_* (For example, MMFS_GENERIC, MMFS_PHOENIX), then this can indicate GPFS is the trigger. Similarly, if any of the fields contain TS_* or RSCT*, then this can indicate that Tivoli SA MP is the trigger. To determine whether this situation occurred, look in the AIX errpt log for an entry similar to the following one:

```
LABEL:          KERNEL_PANIC
IDENTIFIER:     225E3B63

Date/Time:       Mon May 26 08:02:03 EDT 2008
Sequence Number: 976
Machine Id:      0006DA8AD700
Node Id:         coralpib08
Class:           S
Type:            TEMP
Resource Name:   PANIC

Description
SOFTWARE PROGRAM ABNORMALLY TERMINATED

        Recommended Actions
        PERFORM PROBLEM DETERMINATION PROCEDURES
```

  For further details, see the Related Reference.

Troubleshooting

If the affected host is online, run the **db2instance -list** command,. If the **db2instance -list** shows that the member is reported as WAITING_FOR_FAILBACK, look for alerts in the output. Check the alert(s), you might have to clear an alert before the member can fail back to its home host. If there is still no failback, see "A member cannot restart on the home host after a successful restart light" on page 221.

IBM Spectrum Scale filesystem outage with a kernel panic:

A kernel panic has occurred on a member host that is due to a IBM Spectrum Scale trigger. The trigger repeats on a sporadic but recurring basis.

Symptoms

The output of the **db2instance -list** command includes a pending failback operation, as shown in the following example:

```
ID   TYPE    STATE                 HOME_HOST  CURRENT_HOST  ALERT  PARTITION_NUMBER  LOGICAL_PORT  NETNAME
--   ----    -----                 ---------  ------------  -----  ----------------  ------------  -------
0    MEMBER  WAITING_FOR_FAILBACK  hostA      hostB         NO                    0             1  hostB-ib0
1    MEMBER  STARTED               hostB      hostB         NO                    0             0  hostB-ib0
2    MEMBER  STARTED               hostC      hostC         NO                    0             0  hostC-ib0
128  CF      PRIMARY               hostD      hostD         NO                    -             0  hostD-ib0
129  CF      PEER                  hostE      hostE         NO                    -             0  hostE-ib0

HOSTNAME             STATE      INSTANCE_STOPPED   ALERT
--------             -----      ----------------   -----
hostA                INACTIVE   NO                 YES
hostB                ACTIVE     NO                 NO
hostC                ACTIVE     NO                 NO
hostD                ACTIVE     NO                 NO
hostE                ACTIVE     NO                 NO
```

In the previous example, hostA has a state of INACTIVE, and an ALERT field is marked as YES. This output of the **db2instance -list** command is seen when hostA is offline or rebooting. Since the home host for member 0, hostA is offline, member 0 has failed over to hostB. Member 0 is now waiting to failback to its home host, as indicated by the WAITING_FOR_FAILBACK state. After hostA is rebooted from the panic, member 1 will fail back to hostA.

Diagnosis

When you check the db2diag log file, you can find many log entries that indicate that a restart light operation has occurred, as shown in the following example:

```
2009-08-27-23.37.52.416270-240 I6733A457           LEVEL: Event
PID     : 1093874             TID  : 1             KTID : 2461779
PROC    : db2star2
INSTANCE:                     NODE : 000
HOSTNAME: hostB
EDUID   : 1
FUNCTION: Db2, base sys utilities, DB2StartMain, probe:3368
MESSAGE : Idle process taken over by member
DATA #1 : Database Partition Number, PD_TYPE_NODE, 2 bytes
996
DATA #2 : Database Partition Number, PD_TYPE_NODE, 2 bytes
0
```

Another way to diagnose this type of problem is to check the system log. Run the OS command **errpt -a** to view the contents of the AIX errpt system log. In the AIX errpt log, you might see log entries similar in the following example, which is for hostA:

```
LABEL:          KERNEL_PANIC
IDENTIFIER:     225E3B63

Date/Time:       Mon May 26 08:02:03 EDT 2008
Sequence Number: 976
Machine Id:      0006DA8AD700
Node Id:         hostA
Class:           S
Type:            TEMP
Resource Name:   PANIC

Description
SOFTWARE PROGRAM ABNORMALLY TERMINATED

        Recommended Actions
        PERFORM PROBLEM DETERMINATION PROCEDURES

Detail Data
ASSERT STRING
5.1: xmemout succeeded rc=d

PANIC STRING
kx.C:2024:0:0:04A53FA8::advObjP == ofP->advLkObjP
```

If you see a `KERNEL_PANIC` log entry as shown in the previous example, the system reboot might be due to an operating system kernel panic that was triggered by a problem in the IBM Spectrum Scale subsystem. A kernel panic and system reboot can be the result of excessive processor usage or heavy paging on the system when the IBM Spectrum Scale daemons do not receive enough system resources to perform critical tasks. If you experience IBM Spectrum Scale filesystem outages that are related to kernel panics, the underlying processor usage or paging issues must be resolved first. If you cannot resolve the underlying issues, run the **db2support** command for the database with the **-s** parameter to collect diagnostic information and contact IBM Technical Support.

Troubleshooting options for the `db2cluster` command

The advanced help menu of the **db2cluster** command lists a number of options that can help diagnose and troubleshoot issues with the cluster manager or the shared file system cluster. It is recommended that you only use these options with the guidance of Service.

Repairing the cluster manager resource model:

If there are inconsistencies between the cluster manager resource model and the `db2nodes.cfg` file, attempts to start the Db2 pureScale instance or new resources fail. In this situation, you must repair the cluster manager resource model and reissue the **db2start** command.

Before you begin

The Db2 instance must be completely stopped before running this command.

About this task

This task shows you how to repair the cluster manager resource model after it has been detected that there are inconsistencies between the `db2nodes.cfg` file and the cluster manager (usually when **db2start** fails). These inconsistencies occur if any of these situations have occurred:

- The `db2nodes.cfg` file was manually edited
- The resource model was manually edited
- An attempt to add or drop a node failed

The **db2cluster** command is used to remove and re-create the resource model by using information stored in an internal file.

To perform this operation, you must be a user in the SYSADM, SYSCTL, or SYSMAINT group.

Procedure

To repair the cluster manager resource model, issue this command from an online host:

```
db2cluster -cm -repair -resources
```

Results

If the cluster resource model cannot be repaired, contact an IBM Service Representative for more information about how to recover from this problem.

Repairing the cluster manager domain:

If a failure situation occurs with a Db2 pureScale instance which requires the cluster manager domain to be re-created, use the **db2cluster** command to re-create it.

Before you begin

The Db2 instance must be stopped before performing this task; all nodes in the cluster must be online.

About this task

Use the **db2cluster** command to remove and re-create the domain and the resource model of the instance.

The domain is re-created using the same topology and configuration as the existing domain (such as the cluster services tiebreaker, host failure detection time).

Restrictions

The command used in this task can only be run as the Db2 cluster services administrator.

Procedure

1. Use the **DB2INSTANCE** environment variable to specify the target instance.
   ```
   export DB2INSTANCE=<inst-name>
   ```
2. Issue the **db2cluster** command with the `-repair -cluster` option while inside the install directory or the `sqllib/bin` directory.
   ```
   db2cluster -cm -repair -domain <domain-name>
   ```

   To obtain the CM domain name, run the **db2cluster** command: **db2cluster -cm -list -domain**. (You can also obtain the domain name with the **db2greg -dump** command.)

   If there are any resources still online, the **db2cluster** command fails and issues a message indicating that the command must be reissued using the `-force` option.

Results

After successful re-creation of the cluster manager domain, bring the instance back online using the **db2start** command.

If the cluster manager domain cannot be successfully re-created, contact an IBM Service Representative for more information about how to recover from this problem. You might be asked to provide the original cluster manager configuration information, which **db2cluster** saves to a text file (/tmp/ ibm.db2.db2cluster_cm_repar_domain_config_<*TIMESTAMP*>) before it tears down the cluster manager domain.

Example

A DBA with Db2 cluster services authority needs to re-create a cluster manager domain, MYDOMAIN, in Db2 instance MYINST.

```
export DB2INSTANCE=myinst1
```

```
db2cluster -cm -repair -domain mydomain
```

As the domain is torn down and re-created, **db2cluster** issues informational messages about the progress and the successful completion of the operation:

```
Deleting the domain 'mydomain' from the cluster ...
Deleting the domain 'mydomain' from the cluster was successful.
Creating domain 'mydomain' in the cluster ...
Creating domain 'mydomain' in the cluster was successful.
Configuring quorum device for domain 'mydomain' ...
Configuring quorum device for domain 'mydomain' was successful.
The host failure detection time has been set to 8 seconds.
The resource model for the instance 'myinst1' has been re-created.
The cluster manager domain has been successfully repaired.
```

Setting the unhealthy host response:

In rare situations, it is possible that excessive load or paging on a single host in the cluster impacts throughput or availability in the rest of the cluster.

The standard response to this situation is make configuration or hardware changes to increase memory; however, if that is not possible, you can also specify that automatic corrective action is taken, an *unhealthy host response*, to prevent any impact on the cluster.

Before you begin

To perform this task, you need to be the Db2 cluster services administrator.

About this task

By default, Db2 cluster services takes no action if this unhealthy host problem occurs; no information related to this unhealthy host feature is written to the log and you must use other tools to monitor the relevant system specifics related to the unhealthy hosts. This task describes how to specify that when this problem is detected, one of two actions automatically occurs:

reboot the host
: This reboots the host, forcing any member on that host to restart in light mode on another host and any primary cluster caching facility to fail over the primary role to the secondary cluster caching facility. After the reboot is complete, any resident member restarts on that host (unless automatic failback is disabled), and any cluster caching facility restarts as the secondary cluster caching facility. Because this provides the strongest assurance that the issues on the unhealthy host will have the least impact on the rest of the cluster, this is the preferred option.

offline the member or cluster caching facility
: This takes any member or cluster caching facility on the host offline. Any processes on the member are stopped and it will restart in light mode on another host. If the primary cluster caching facility is on that host, the secondary cluster caching facility will take over the primary role. An alert is raised that needs to be cleared manually for the member to failback or for the cluster caching facility to start again. This option is suitable if there are other things on the host that you do not want to be impacted or if you want to keep the host available for diagnosing the problem. The manual clearing of the raised alert is also a possible reason for choosing this option.

It bears repeating that setting this option is really a last-resort response for preventing unplanned cluster issues. If you suspect that this unhealthy host state might occur, the best response is to add memory resources to the host.

Procedure

To set up the unhealthy host response:

- If you want to automatically reboot the host, issue the following command:

  ```
  db2cluster -cm -create -unhealthy_host_response -option -reboot_host -option
  -apply_to_current_host
  ```

- If you want to automatically take any member or cluster caching facility on the host offline, issue the following command:

  ```
  db2cluster -cm -create -unhealthy_host_response -option -offline_member -option
   -apply_to_current_host
  ```

Results

If you specified host reboot as the automatic response and the host is successfully rebooted, then any member should be restarted on the host. After a reboot, the member should be restarted on the host, unless automatic failback is disabled. Any cluster caching facility on the host should be restarted on the host, but if it was previously the primary cluster caching facility on the host , it will now be the secondary cluster caching facility on the host. Information about the reboot event is written to the **syslog** file. For more information, see the "Related links" section.

If you specified that the member or cluster caching facility should be taken offline and this does occur, the member or cluster caching facility will have an alert on it. The offlined member will not restart on or failback to its home host until this alert is removed. The offlined cluster caching facility will not restart until this alert is removed

Uninstallation

This section contains information that will help you understand and resolve problems that you might encounter while uninstalling the IBM Db2 pureScale Feature.

Cleaning up an incomplete Db2 pureScale instance drop:

When the **db2idrop** command has failed to completely drop the instance, you must complete the clean up of a Db2 pureScale instance.

Procedure

To clean up an incomplete Db2 pureScale instance drop:

1. Remove the RSCT Peer Domain (PD) forcefully:
   - Try to run `rmrpdomain -f` *domain_name*.

   If you do not know what the domain name is, run **lsrpdomain** and look for the domain name under the Name column.

   ```
   $ lsrpdomain
   Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
   db2domain Online  2.5.3.5           No            12347  12348
   ```
2. Clean up the Db2 cluster file system.
3. Clean up `$HOME/sqllib` for the instance user on all hosts.
4. Clean up the entries in `/etc/services` for the Db2 pureScale instance user.
5. Remove instance signature

   installed_path/instance/db2iset -d *instance_name*
6. Remove following global registries on all hosts:

- `INSTPROF`
- `PEER_DOMAIN`

For example

```
installed_path/bin/db2greg -delvarrec
service=GPFS_CLUSTER,variable=NAME,installpath=-
```

```
installed_path/bin/db2greg -delvarrec
service=PEER_DOMAIN,variable=NAME,installpath=-
```

```
installed_path/bin/db2greg -delvarrec
service=DEFAULT_INSTPROF,variable=DEFAULT,installpath=-
```

```
installed_path/bin/db2greg -delvarrec
service=INSTPROF,variable=<instance name>,installpath=-
```

Cleaning up an IBM Reliable Scalable Cluster Technology peer domain:

This topic will guide you through the required steps to manually clean up a Reliable Scalable Cluster Technology (RSCT) peer domain. Uninstalling Db2 Enterprise Server Edition with IBM Db2 pureScale Feature will normally handle the removal of RSCT. Follow these steps if an unexpected error occurs and a manual cleanup is required.

About this task

The **db2idrop -g** command should automatically remove the IBM Reliable Scalable Cluster Technology (RSCT) peer domain. The following manual steps are needed only when problems occur.

Procedure

1. List all of the existing peer domains by running the **lsrpdomain** command. The output of this command should be similar to the following one:

   ```
   lsrpdomain
   ```

   If nothing is listed, there is no active peer domain. Only one RSCT peer domain can be active (online) at any given time, and any operations (stopping, removing, or adding a node) will only affect this online domain.
2. List all of the nodes in this active peer domain by running the **db2cluster -cm -list -host -state** command. The output of this command should be similar to the following output:

   ```
   HOSTNAME    STATE
   ----------- ------
   coralpib135 ONLINE
   coralpib136 ONLINE
   ```
3. Remove the entire peer domain, including all nodes. The peer domain must be online for the following command to be successful. Run the **db2cluster -cm -delete -domain db2domain** command. The output of this command should be similar to the following output:

   ```
   Deleting the domain db2domain from the cluster ...
   Deleting the domain db2domain from the cluster was successful.
   ```
4. To confirm that the peer domain has been removed appropriately, run the **db2cluster -cm -list -domain** command.

Frequently asked questions about uninstallation problems with the Db2 pureScale Feature:

Use this topic to determine possible solutions to problems with uninstalling the IBM Db2 pureScale Feature.

What if the db2idrop command failed to remove the host from the GPFS cluster?

To help determine the cause of the failure, review the **db2cluster** command log file in the *DB2DIR*/tmp/ibm.db2.cluster.* directory. After resolving the problem, reissue the **db2idrop** command.

What if the db2idrop command failed because the instance is not usable?

To help determine the cause of the failure, review the **db2idrop** command log in the *DB2DIR*/tmp directory. Check if instance_user/sqllib/db2nodes.cfg is valid.

What if the db2idrop command failed when removing the GPFS CLUSTER global registry (or the PEER DOMAIN global registry) on specific hosts?

If this failure occurs, contact IBM Software Support.

What if the db2idrop command failed to remove the RSCT peer domain?

If the **db2idrop** command failed to remove the RSCT peer domain, you will have to manually remove it by following these steps:

1. Check if there is still resource attached to the peer domain by running the **lssam** command.[4]
2. If there are still resources attached, perform the following steps:
   a. Switch to the instance owner by entering `su - instance_owner`.
   b. Remove the resource by entering `db2cluster -cm -delete -resources`.
   c. Switch back to root.
3. Remove RSCT peer domain by running the `db2cluster -cm -delete -domain domain_name` command from the DB2DIR/bin directory. Run the `lsrpdomain` command to determine the domain name to specify.

For more information, see Manually cleaning up an IBM Reliable Scalable Cluster Technology peer domain.

What if the db2_deinstall command failed?

The **db2_deinstall** command can uninstall Db2 database products, features, or languages on the host where the **db2_deinstall** command is run. The **db2_deinstall** command might fail because the host is still part of the GPFS cluster or the host still belongs to the RSCT peer domain. If the host is still part of the GPFS cluster, run the `db2_deinstall -s GPFS` command to remove it.

If there still exists a Db2 pureScale instance, you can use the `db2iupdt -drop` or `db2idrop -g` command to remove the Db2 pureScale instance. Then rerun the **db2_deinstall** command.

If you want to keep the peer domain, you can bypass the IBM Tivoli System Automation for Multiplatforms binary file removal by running the `db2_deinstall -a -s TSAMP`. If there are no resources using the peer domain on this host, before reissuing the command, manually remove the Tivoli SA MP.

4. For more information on Tivoli SA MP commands, see this https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Tivoli%20Documentation%20Central/page/Tivoli%20System%20Automation%20for%20Multiplatforms.

Troubleshooting Db2 Connect Servers

The Db2 Connect environment involves multiple software, hardware and communications products. Troubleshooting is best approached by a process of elimination and refinement of the available data to arrive at a conclusion (the location of the error).

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

Diagnostic tools

Db2 Connect provides diagnostic tools to troubleshoot problems. You can also use the tools and diagnostic files provided with the operating system.

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

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

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

      Note: The directory does not have to be named SQLLIB.
    - If the Db2 registry variable **DB2INSTPROF** is set then `x:\DB2INSTPROF\DB2INSTANCE` is used where `x:\DB2INSTPROF` is the path specified in the **DB2INSTPROF** registry variable and **DB2INSTANCE** is the name of the instance (by default, the value of **DB2INSTDEF** on Windows 32-bit operating systems).
- For Windows operating systems, you can use the Event Viewer to view the administration notification log.
- The available diagnostic tools that can be used include **db2trc**, **db2pd**, **db2support** and **db2diag**
- 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.

Gathering relevant information

Troubleshooting includes narrowing the scope of the problem and investigating the possible causes. The proper starting point is to gather the relevant information and determine what you know, what data has not been gathered, and what paths you can eliminate.

At a minimum answer the following questions.

- Has the initial connection been successful?
- Is the hardware functioning properly?

- Are the communication paths operational?
- Have there been any communication network changes that would make previous directory entries invalid?
- Has the database been started?
- Is the communication breakdown between one or more clients and the Db2 Connect Server (gateway); between the Db2 Connect gateway and the IBM mainframe database server
- What can you determine by the content of the message and the tokens returned in the message?
- Will using diagnostic tools such as **`db2trc`**, **`db2pd`**, or **`db2support`** provide any assistance at this time?
- Are other machines performing similar tasks working correctly?
- If this is a remote task, is it successful if performed locally?

Initial connection is not successful

If you have configured a new connection in Db2 Connect and cannot connect successfully, troubleshoot the problem by answering a set of questions that are structured into a checklist.

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 IBM mainframe database server and workstation communications configured properly?*
   - There are three configurations that must be considered:
     a. The IBM mainframe database server configuration identifies the application requester to the server. The IBM mainframe server database management system will have system catalog entries that will define the requester in terms of location, network protocol and security.
     b. The Db2 Connect workstation configuration defines the client population to the server and the IBM mainframe 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 IBM mainframe server database administrator and the Network administrators have utilities available to diagnose problems.

5. *Do you have the level of authority required by the IBM mainframe server database management system to use the IBM mainframe 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 IBM mainframe database server, are you unsuccessful?*
   - Did you follow the procedure to bind the CLP to the IBM mainframe database server?

If the checklist does not guide you to a resolution, contact IBM Support.

Problems encountered after an initial connection

If Db2 Connect can no longer connect successfully, troubleshoot the problem by answering a set of questions that are structured into a checklist.

Answering the following questions can help you to identify the source of the connection problem:

1. *Are there any special or unusual operating circumstances?*
   - Is this a new application?
   - Are new procedures being used?
   - Are there recent changes that might be affecting the system? For example, have any of the software products or applications been changed since the application or scenario last ran successfully?
   - For application programs, what application programming interface (API) was used to create the program?
   - Have other applications that use the software or communication APIs been run on the user's system?
   - Has a fix pack recently been installed? If the problem occurred when a user tried to use a feature that had not been used (or loaded) on their operating system since it was installed, determine IBM's most recent fix pack and load it *after* installing the feature.
2. *Has this error occurred before?*
   - Are there any documented resolutions to previous error conditions?
   - Who were the participants and can they provide insight into a possible course of action?
3. *Have you explored using communications software commands that return information about the network?*
   - TCP/IP might have valuable information retrieved from using TCP/IP commands and daemons.
4. *Is there information returned in the SQLCA (SQL communication area) that can be helpful?*
   - Problem handling procedures should include steps to examine the contents of the SQLCODE and SQLSTATE fields.
   - SQLSTATEs allow application programmers to test for classes of errors that are common to the Db2 family of database products. In a distributed relational database network this field might provide a common base.
5. *Was START DBM executed at the Server?* Additionally, ensure that the **DB2COMM** environment variable is set correctly for clients accessing the server remotely.
6. *Are other machines performing the same task able to connect to the server successfully?* The maximum number of clients attempting to connect to the server might

have been reached. If another client disconnects from the server, is the client who was previously unable to connect, now able to connect?

7. *Does the machine have the proper addressing?* Verify that the machine is unique in the network.
8. *When connecting remotely, has the proper authority been granted to the client?* Connection to the instance might be successful, but the authorization might not have been granted at the database or table level.
9. *Is this the first machine to connect to a remote database?* In distributed environments routers or bridges between networks might block communication between the client and the server. For example, when using TCP/IP, ensure that you can PING the remote host.

Unsupported DDM commands:

The DDM commands **BNDCPY**, **BNDDPLY**, **DRPPKG** and **DSCRDBTBL** are not supported by Db2 Version 9.5 for Linux, UNIX, and Windows when it is acting as a DRDA application server (DRDA AS).

Symptoms

If a DRDA application requester (DRDA AR) connects to Db2 Version 9.5 for Linux, UNIX, and Windows and issues any of the following commands, the command will fail:

Table 11. Unsupported DDM commands

DDM command	DDM code point	Description
BNDCPY	X'2011'	Copy an existing relational database (RDB) package
BNDDPLY	X'2016'	Deploy an existing RDB package
DRPPKG	X'2007'	Drop a package
DSCRDBTBL	X'2012'	Describe an RDB table

In addition, the following code points, used in the SQLDTA descriptor for parameter-wise (or column-wise) array input, are also not supported:

Table 12. Unsupported FD:OCA data objects

FD:OCA data objects	DDM code point	Description
FDOEXT	X'147B'	Formatted Data Object Content Architecture (FD:OCA) Data Extents
FDOOFF	X'147D'	FD:OCA Data Offsets

The most common error message in this situation is SQL30020N ("Execution failed because of a Distributed Protocol Error that will affect the successful execution of subsequent commands and SQL statements").

Causes

Distributed Data Management Architecture (DDM) is part of the DRDA protocol. The DDM commands **BNDCPY**, **BNDDPLY**, **DRPPKG** and **DSCRDBTBL** exist in all of the

DRDA levels that are supported by Db2 Version 9.5 for Linux, UNIX, and Windows but the DRDA application server does not support these DDM commands.

Likewise, a Db2 Version 9.5 for Linux, UNIX, and Windows DRDA application server does not support the **FDOEXT** and **FDOOFF** code points. These code points are used in the SQLDTA descriptor that is sent to server when you submit a column-wise array input request.

Diagnosing the problem

If you obtain a Db2 trace on the DRDA application server, you will see a message similar to the following in response to these commands:`ERROR MSG = Parser: Command Not Supported`.

Resolving the problem

There are currently no supported alternatives for the **BNDCPY** and **BNDDPLY** DDM commands.

To drop a package, use the SQL statement DROP PACKAGE. For example, connect to the Db2 Version 9.5 for Linux, UNIX, and Windows DRDA application server and send a DROP PACKAGE statement in an EXECUTE IMMEDIATE request. Db2 Version 9.5 for Linux, UNIX, and Windows will process that request successfully.

To describe an RDB table, use one of the following DDM commands: **DSCSQLSTT** (Describe SQL Statement) or **PRPSQLSTT** (Prepare SQL Statement). For example, if you want a description of the table TAB1, describe or prepare the following statement: `SELECT * FROM TAB1`.

Note: When the DRDA AR issues the **PRPSQLSTT** command, it is necessary to also specify the instance variable **RTNSQLDA** with a value of `TRUE`, otherwise the SQLDA Reply Data (SQLDARD) descriptor will not be returned by the server.

To avoid problems with the **FDOEXT** and **FDOOFF** code points, use row-wise array input requests instead of parameter-wise (or column-wise) array input requests.

Common Db2 Connect problems

There are common symptoms and solutions for connection problems that you can encounter when using Db2 Connect.

In each case, you are provided with:

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

SQL0965 or SQL0969

Symptom

Messages SQL0965 and SQL0969 can be issued with a number of different return codes from IBM Db2 for IBM i, Db2 for z/OS, and Db2 Server for VM and VSE.

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

Solution

The SQL code received from the IBM mainframe database cannot be translated. Correct the problem, based on the error code, then resubmit the failing command.

SQL5043N

Symptom

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

Perhaps the TCP/IP protocol is not started on the Db2 Connect server. There might have been a successful client connection previously.

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

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

Solution

This warning is a symptom which signals that Db2 Connect, acting as a server for remote clients, is having trouble handling one or more client communication protocols. These protocols can be TCP/IP and others, and usually the message indicates that one of the communications protocols defined to Db2 Connect is not configured properly.

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

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

Most likely, one of the clients connecting to the Db2 Connect Server still has a handle on the TCP/IP port. On each client machine that is connected to the Db2 Connect Server , enter the following commands:

```
db2 terminate
db2stop
```

SQL30020

Symptom

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

Solutions

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

SQL30060

Symptom

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

Solution

When connecting to Db2 for z/OS, the Communications Database (CDB) tables have not been updated properly.

SQL30061

Symptom

Connecting to the wrong IBM mainframe database server location - no target database can be found.

Solution

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

Check the Db2 node, database, and DCS directory entries. The target database name field in the DCS directory entry must correspond to the name of the database based on the platform. For example, for a Db2 for z/OS database, the name to be used should be the same as that used in the Boot Strap Data Set (BSDS) "LOCATION=*locname*" field, which is also provided in the DSNL004I message (LOCATION=*location*) when the Distributed Data Facility (DDF) is started.

The correct commands for a TCP/IP node are:

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

To connect to the database you then issue:

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

SQL30081N with Return Code 79

Symptom

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

Solution(s)

This error can occur in the case of a remote client failing to connect to a Db2 Connect Server. It can also occur when connecting from the Db2 Connect Server to a IBM mainframe database server.

1. The **DB2COMM** profile variable might be set incorrectly on the Db2 Connect Server. Check this. For example, the command `db2set db2comm=tcpip` should appear in `sqllib/db2profile` when running Db2 Enterprise Server Edition on AIX.

2. There might be a mismatch between the TCP/IP service name and port number specifications at the IBM data server client and the Db2 Connect Server. Verify the entries in the TCP/IP `services` files on both machines.
3. Check that Db2 is started on the Db2 Connect Server. Set the Database Manager Configuration **diaglevel** to 4, using the command:

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

   After stopping and restarting Db2, look in the **db2diag** log files to check that Db2 TCP/IP communications have been started. You should see output similar to the following one:

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

SQL30081N with Protocol Specific Error Code 10032

Symptom

```
SQL30081N  A communication error has been detected.
Communication protocol
being used: "TCP/IP".  Communication API being used: "SOCKETS".
Location
where the error was detected: "9.21.85.159".  Communication
function detecting
the error: "send".  Protocol specific error code(s): "10032",
"*", "*".
SQLSTATE=08001
```

Solution

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

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

SQL30082 RC=24 During CONNECT

Symptom

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

Solution

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

Troubleshooting Db2 Text Search

Troubleshooting and maintenance tasks for Db2 Text Search include, for example, deleting orphaned text search collections and adjusting log and trace properties.

There are three resources that can help you perform problem determination for Db2 Text Search: the event table, the **db2trc** command, command, and the log and trace facilities of the Db2 Text Search server.

Using the Db2 trace facility for text search operations

If you need to report an error to an IBM representative, you might be asked to switch on tracing so that information can be written to a file that can be used for locating the error.

About this task

You can use the Db2 tracing facility, **db2trc**, to capture information about the problems that might occur while you are performing text searches or administrative operations associated with text searches.

Because system performance is affected when tracing is switched on, use the trace facility only when directed by an IBM Support Center representative or by your technical support representative.

To turn tracing on and receive information specific to Db2 Text Search, run the **db2trc** command using a mask with a component code for CIE (155), as follows:

```
db2trc on -m '*.*.CIE.*.*'
```

To help diagnose severe errors, it might also help to look in the **db2diag** log file.

Additional tracing facilities are available for the Text Search server. For details, see the topic about logging and tracing for the Db2 Text Search server.

Logging and tracing for the Db2 Text Search server

The Db2 Text Search server logs system messages and traces messages to help you determine the source of problems that might occur. Diagnostic data that you collect, and the sources from which you collect that data, depend on the type of problem that you are investigating.

For specific symptoms, you might have to collect additional problem-specific data from the following locations:

- *<install-path>* for the server. This might vary depending on the server type:
  - *<DBPATH>/db2tss* for the integrated text search server
  - *<ECMTS_HOME>* for the stand-alone text search server
- *<absolute-path-to-config-folder>* the location for configuration files for the text search server
- *<collection-path>* the directory that is specified as COLLECTION DIRECTORY in the CREATE INDEX operation (only for integrated text search servers), or if the default is used, in the defaultDataDirectory (for integrated and stand-alone text search servers).

To collect diagnostic data for the stand-alone Db2 Text Search server:

1. Run the administration tool with the version command:
   ```
   adminTool version -configPath <absolute-path-to-config-folder>
   ```
2. Collect log files that are stored in the `logpath` configured for the text search server.
3. In some cases, you might want to capture more detailed information:
   - Specify your preferred command-line tool logging properties by editing the `ecmts_config_logging.properties` file in the directory *<absolute_path_to_config_folder>*.
   - Specify trace logging properties by editing the `ecmts_logging.properties` file in the directory *<absolute_path_to_config_folder>*.

- Enable tracing by using the adminTool command.

```
adminTool configureTrace -configPath <absolute-path-to-config-folder> -trace on
```

4. Gather the following configuration files from the <*absolute-path-to-config*> directory:
   a. `configuration.xml`
   b. `key.txt`
   c. `authentication.xml`
   d. `constructors.xml`
5. If you encounter a collection-related issue when creating, deleting, indexing, or searching a collection, locate the collection configuration file in the <*collection-path*> directory, and if required, collection data from the <*collection-path*>`/data` directory.
6. If you encounter an issue that is related to stop words, collect the stop word XML files from the <*install-path*>`/resource/uima` folder.

   Stop word XML files are named as <*language_code*>`-Stw.xml`.
7. If you encounter an issue that is related to synonyms, collect the synonyms from the <*collection-path*>`/data/synonym` directory and contact Db2 Support.

Monitoring queues for Db2 Text Search index updates

You can gather monitoring information to tune the Db2 Text Search server configuration if you experience indexing performance issues.

You can monitor input and output queues by starting the Db2 Text Search server with the -monitorQueues flag. When you start the server with the -monitorQueues flag, information about the current state of the input and output queues is printed to the `InputQueueSizes.csv` and `OutputQueueSizes.csv` files. These files are stored in the *ECMTS_HOME*`\logs` directory. Each file provides the following information:

- The time.
- The current number of documents that are waiting to be preprocessed and indexed.
- The number of documents that are currently being preprocessed and indexed.
- The current queue size. The maximum queue size is a configurable parameter.
- The total number of documents that passed through the queue, that is, the number of documents that were preprocessed and indexed since the last server startup.
- The amount of heap memory in MB that is used by the JVM before Java memory garbage collection.
- The number of threads that are used by the Db2 Text Search server.
- An indication of the average system load for the previous minute and free physical memory, as provided by JVM.
- The number of open operating system file descriptors. This information is available only for AIX, Linux, and Solaris systems.
- The number of index segment merges that are currently taking place.
- The total size (in MB) of index segment merges that are currently taking place.

Important: The CSV files are re-created each time that the server is started. If you want to store the information in these files, back them up before you restart the server.

To start the server in monitor queue mode:

1. Stop the server.

2. Edit the startup script in the *ECMTS_HOME*\bin directory.
3. Add the -monitorQueues flag to the startup command. For example:
   ```
   "%JAVA_HOME%\bin\java" -classpath "%CLASSPATH%"
   com.ibm.es.nuvo.launcher.Launcher "%CONFIG_XML%" -monitorQueues
   -Xmx%HEAP_SIZE% %1 %2 %3 %4 %5 %6 %7 %8 %9 %IPv6Flag% %JVM_OPTIONS%
   ```
4. Save the startup script and restart the server.

Use the information in the following table to troubleshoot indexing performance that is based on queue status.

Table 13. Troubleshooting indexing performance based on queue status

Queue status	Troubleshooting strategy
Input queue is empty	Documents are not being received from the client. Ensure that the client is pushing documents fast enough to the server.
Input queue is full, output queue is empty	Check the logs for preprocessing errors and consider increasing the number of preprocessing threads. Low CPU levels (less than 60%) on servers in a large multiprocessor system can also indicate that the number of preprocessing threads must be increased. The number of preprocessing threads must not exceed the number of processors. In a distributed architecture, ensure that each preprocessing server is running and is configured correctly. For example, verify that the connection parameters (indexing server host name, port number, and token) are configured correctly on each preprocessing server. You can check the connectivity by monitoring the input queue on each preprocessing server to ensure that documents are received. You can also try to increase the number of preprocessing threads on each preprocessing server.
Both input and output queues are full	Try one or more of the following strategies: • Increase the number of index threads. • Configure more frequent, shorter merges by reducing the value of the **MaxMergeDocs** parameter in the *ECMTS_HOME*\config\collections\collection_name\collection.xml file. • Increase the batch size. Indexing documents in small batches can degrade performance, because the index is flushed and saved to disk for every batch that is indexed. • Increase the **BufferSize** parameter in the *ECMTS_HOME*\config\collections\collection_name\collection.xml file. • Index one collection at a time. • Ensure that the disk speed and network connection is fast enough.

Stand-alone text search server fails to start

If the stand-alone text search server fails to start when you try starting it using the `startup.bat` startup script, set the port number to a new value and try starting the stand-alone server again.

Symptoms

When you try starting the standalone text search server by running the startup.bat script, the server might fail to start and you might see the following error message:

```
IQQD0056E An error occurred when starting the server at port 8191.
IQQG0335E The server cannot start. See the error log file for more details.
ExtendedException: D0056E.CONNECT_ERROR - D0056E.CONNECT_ERROR
```

You can also check for a similar error message in the *ecmts_install_directory*/`log/trace0.log` log file. The error message in this log file looks like as follows:

```
IQQD0056E An error occurred when starting the server at port 8191.
Causes of the problem:
  IQQG0020E java.net.SocketException: Unrecognized Windows Sockets error: 0: JVM_Bind
      java.net.PlainSocketImpl.socketBind(Native Method)
      java.net.PlainSocketImpl.bind(PlainSocketImpl.java:415)
      java.net.ServerSocket.bind(ServerSocket.java:330)
      java.net.ServerSocket.(ServerSocket.java:196)
      java.net.ServerSocket.(ServerSocket.java:152)
      com.ibm.es.nuvo.inyo.common.InyoServer.run(InyoServer.java:188)
```

Resolving the problem

To resolve this error message, perform either of the following steps:

- Set the port number to a new value by issuing the following command:`configTool configureParams -adminHTTPPort` *new_port_value* `-configPath` *ecmts_install_directory*`\config\config.xml`.

  After setting the port number to a new value, try running the startup.bat script again.
- Uninstall the standalone server and then reinstall the server by setting the port number to a new value that is defined in the `ecmts_response.txt` response file.

Troubleshooting hints and tips for Db2 Text Search

Troubleshooting your database requires knowledge of what tools to use and how to properly use the tools to diagnose problems.

Use the information in the following table to help troubleshoot problems.

Table 14. Issues and solutions

Issue	Solution
If you drop and re-create a text index while a Db2 connection session is active, a plan for a previously used query (for Net Search Extender or Db2 Text Search) might be cached and reused, thus producing incorrect results. If the Db2 Text Search engine encounters this issue, IQQG0037W will be returned, indicating that an incorrect collection name was used for the search.	Use the FLUSH PACKAGE CACHE DYNAMIC statement to remove the execution plans for the SQL statements from the statement cache.
If you drop or update your Db2 instance using the **db2idrop**, **db2iupdt**, or **db2iupgrade** command, the entire `sqllib` directory in the instance is removed, including the `sqllib/db2tss/config` directory. This will make it hard to reuse text search instance databases.	Back up the `sqllib/db2tss/config` directory before issuing the **db2idrop**, **db2iupdt**, or **db2iupgrade** command. Also, use the **printAll** option of the **configTool** command to save a record of the configuration parameters, which should help you to configure the new instance.

Table 14. Issues and solutions (continued)

Issue	Solution
A disk full error might cause the text index metadata to be inconsistent.	Use the Administration Tool to check whether there are any orphaned collections and to remove them, as outlined in "Deleting orphaned Text Search collections" in the Db2 Text Search documentation. Note that you might not be able to reuse the text index name that was being used when you encountered the disk full error.
You receive the message CIE00301 "Insufficient authority" or CIE00377 "Only instance owner <*instance-owner*> can use that command" after issuing a db2ts **`START FOR TEXT`** command.	For Windows users, have a user with sufficient authorization issue db2ts **`START FOR TEXT`**. For Linux and UNIX, have the instance owner issue this command.
If the Text Search server encounters a disk full error, it might return the following error message: `IQQD0085E The requested operation cannot be done. The server is running in safe mode due to index has an IO error.`	The Db2 Text Search server is running in a restricted mode. Shutdown the text search server, and restart it after resolving the disk space issue.
You receive the message CIE00311 telling you that an internal file cannot be opened. This message can indicate a missing configuration directory, or indicate that a file may have been lost or corrupted, for example, due to a disk full error on the filesystem where `db2tss/config` is located, or because of a problem during the backup of the `db2tss/config` directory.	Check to see if the instance has a `config` directory for text search. • If the `config` directory is missing ensure that Db2 Text Search was installed and configured. • If the `config` directory is in another location, add a symbolic link to it (UNIX). If this error is being caused by a missing or corrupted file, contact IBM Support.

You receive the message CIE00445N telling you that the requested operation cannot be executed. Run the **`REBIND`**. This message can indicate that one of the previous Db2 commands has invalidated packages "NULLID.SYSSH100, NULLID.SYSSH200, NULLID.SYSSN100, NULLID.SYSSN200 or NULLID.SYSLH202" which is required for Db2 Text Search.	User needs to manually rebind the package "NULLID.SYSSH100, NULLID.SYSSH200, NULLID.SYSSN100, NULLID.SYSSN200 or NULLID.SYSLH202". before continuously executing db2ts commands. The package invalidation usually occurs when revoke or grant commands are executed.
	User can prevent running into this error by checking the packages state through the following SQL statement:
	`select 1 from syscat.packages where`
	`VALID = 'N' AND`
	`((PKGSCHEMA='NULLID' AND PKGNAME=`
	`'SYSSH100') OR`
	`(PKGSCHEMA='NULLID' AND PKGNAME=`
	`'SYSSH200') OR`
	`(PKGSCHEMA='NULLID' AND PKGNAME=`
	`'SYSSN100') OR`
	`(PKGSCHEMA='NULLID' AND PKGNAME=`
	`'SYSSN200') OR`
	`(PKGSCHEMA='NULLID' AND PKGNAME=`
	`'SYSLH202'))`
	`FETCH FIRST 1 ROWS ONLY FOR READ ONLY;`
	If the return value is 1 through this SQL statement, user needs to rebind the packages NULLID.SYSSH100, NULLID.SYSSH200, NULLID.SYSSN100, NULLID.SYSSN200 or NULLID.SYSLH202 before executing Db2 Text Search commands.

A text index update with a large number of documents to process fails with an 'insufficient memory' message in the `db2diag.log` file.	If it is not feasible to increase available memory, decrease the `documentqueueresultsize` value in `sysibmts.tsdefaults` administrative view and try again.

Table 14. Issues and solutions (continued)

Issue	**Solution**
You encounter message IQQG0037W message in a query about a missing collection after a data redistribution.	Ensure that the FOR DATA REDISTRIBUTION option is used the next time a text search UPDATE INDEX command is issued.

Troubleshooting RDF

This troubleshooting section contains information that will help you understand and resolve problems that are specific to the Resource Description Framework (RDF).

Troubleshooting performance issues with SPARQL queries

You can take a number of steps to improve the performance of SPARQL queries that take a long time to run.

Use the following information to help improve the performance of SPARQL queries:

- Ensure that you have the **AutoRunStats** database configuration parameter set to `ON`.

  If you recently performed a significant update of triples in the RDF store, consider explicitly invoking the **RUNSTATS** command on the tables of the RDF store. **RUNSTATS** profiles are created for each of the tables in the RDF stores. Invoke **RUNSTATS** on each of the RDF store table with the following command: **db2 RUNSTATS ON** *SCHEMA.TABLE* `USE PROFILE`
- Ensure that you have bound and using the `REOPT ONCE` or `REOPT ALWAYS` packages. Since parameterized SQL statements are used during SQL query execution, you can take advantage of any skew in the data distribution by performing the following steps:
  1. Bind the NULLIDR1 package set with the `REOPT ONCE` bind option or bind the NULLIDRA package set with the `REOPT ALWAYS` bind option.
  2. Specify a correct value for the currentPackageSet property of the DB2BaseDataSource class using the `setXXX` method. In an application server environment, you can set the currentPackageSet property on a `DataSource`.
- Ensure that you have set the schedule for the Db2 Administrative Task Scheduler to gather statistics specific to the RDF stores using the **setstatsschedule** command. Further, ensure that the task is running successfully at the setup time interval. You can do this by first getting the ID of this task from the ADMINTASKS table and then checking the status of this task from the ADMINTASKSTATUS table.

  If you recently performed a significant update of triples in the RDF store, issue the **updaterdfstorestats** command to immediately force the RDF store specific statistics to get updated.
- After performing the previous three steps, if you still experience performance issues with SPARQL queries where the queries take a long time to execute or you encounter any exception during the processing of SPARQL queries, turn on the query log in the RDF store API. The following example shows different methods of turning on the query log in the RDF store API.

```
Store store = StoreManager.connectStore(conn, "db2admin",
 "Sample");

//set the file where sparql query information should be logged
// note if the file already exists, it is overwritten.
store.setQueryLogFile("\querylogfile.txt");
```

```
//set the file where sparql query information should be logged
// and also the time limit (in millsecs). If a query takes more // time that the specified valu
store.setQueryLogFile("\querylogfile.txt",3000);
```

All exceptions that occur during the processing of SPARQL queries are logged in to the query log file. To help resolve these performance and processing issues of SPARQL queries, contact IBM Support and provide the query log file to aid diagnosis..

- The Db2 RDF component throws Java Runtime exceptions whenever an error is encountered. Turn on the LOG4J logging from the Java virtual machine (JVM) application to collect more information during such exceptions. The JVM application can examine the SQL exceptions and error codes in case the problem is a database setup issue.

Getting fixes

A product fix might be available to resolve your problem. You can get fixes by following these steps.

Procedure

1. You can view fix lists and obtain fix packs from the following Web pages:
   - IBM Support Portal: Downloads
   - Fixes by version for DB2 for Linux, UNIX, and Windows
2. Determine which fix pack you need. In order to avoid encountering problems caused by software defects already known and corrected, the installation of the most recent fix pack is generally recommended.
3. Download the fix pack and extract the files by double-clicking the self-extracting executable package. Open the `SERVER/doc/`*`your_language`*`/readme.txt` document and follow the provided link to the *Db2 Information Center* to obtain installation instructions.
4. Apply the fix. For instructions, see: "Applying fix packs" in *Installing Db2 Servers*.

Fix packs, interim fix packs and test fixes

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

The modified Db2 code that resolves the problem described in the APAR can be delivered in fix packs, interim fix packs and test fixes.

Fix pack

A fix pack is a cumulative collection of APAR fixes. In particular, fix packs address the APARs that arise between new releases of Db2. They are intended to allow you to move up to a specific maintenance level. Fix packs have the following characteristics:

- They are cumulative. Fix packs for a particular release of Db2 supersede or contain all of the APAR fixes shipped in previous fix packs and interim fix packs for that release.
- They are available for all supported operating systems and Db2 database products.
- They contain many APARs.

- They are published on the Db2 Technical Support website and are generally available to customers who have purchased products under the Passport Advantage® program.
- They are fully tested by IBM.
- They are accompanied by documentation that describes changes to the database products, and how to install and remove the fix pack.

Note: The status of an APAR is changed from "Open" to "Closed as program error" when the APAR fix is provided in a fix pack. You can determine the status of individual APARs by examining the APAR descriptions on the Db2 Technical Support website.

Interim fix pack

An interim fix pack is a cumulative collection of important APAR fixes that arise between fix packs. To qualify for inclusion in an interim fix pack, an APAR must be considered pervasive or otherwise important. Candidate APARs are evaluated and approved by experts on the Db2 technical support team. Interim fix packs have the following characteristics:

- They are cumulative. Interim fix packs for a particular release of Db2 supersede or contain all of the APAR fixes shipped in previous fix packs and interim fix packs for that release.
- They are available for a subset of operating systems and Db2 database products.
- They usually contain 20 to 30 new APARs.
- They are published on the Db2 Technical Support website and are generally available to customers who have purchased products under the Passport Advantage program.
- They are fully tested by IBM.
- They are accompanied by documentation that describes how to install and remove the interim fix pack.

Interim fix packs are supported in production for two years after they are released. They are available at the approximate midpoint between fix packs and are intended as the preferred alternative to test fixes, which do not receive the same level of testing or enjoy the same level of support as interim fix packs.

Test fix

A test fix is a temporary solution that is supplied to specific customers for testing in response to a reported problem. Test fixes are sometimes referred to as "special builds" and they have the following characteristics:

- They usually contain a single APAR.
- They are obtained from Db2 Support and are not generally available to the public.
- They undergo limited IBM testing.
- They include minimal documentation, including a description of how the test fix should be applied, the APARs it fixes, and instructions for the removal of the test fix.

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

It is recommended that you keep your Db2 environment running at the latest fix pack level to ensure problem-free operation. To receive notification of the availability of new fix packs, subscribe to "My notifications" email updates on the Db2 Technical Support website at: http://www.ibm.com/software/data/db2/support/db2_9/

To learn more about the role and purpose of Db2 fixes and fix packs, see the support policy statement. To help ensure problem-free operation, keep your Db2 environment running at the latest fix pack level. To be notified of the availability of new fix packs, subscribe to "My notifications" email updates on the Db2 Technical Support website (http://www.ibm.com/software/data/db2/support/db2_9/). For a list of fix packs, see the Download Db2 Fix Packs by version for Db2 page (http://www-01.ibm.com/support/docview.wss?uid=swg27007053).

Applying test fixes

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

Before you begin

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

About this task

There are two types of test fixes:

- A test fix for an individual Db2 product. These test fixes can be applied on an existing installation of the product, or can be used to perform a full product installation where there is no existing Db2 installation.
- Universal test fixes (Linux and UNIX only). A universal test fix services installations where more than one Db2 product has been installed.

If national languages have been installed, you might also require a separate national language test fix. The national language test fix can only be applied if it is at the same test fix level as the installed Db2 product. If you are applying a universal test fix, you must apply both the universal test fix and the national language test fix to update the Db2 products.

Procedure

Obtain the test fix from IBM Software Support and follow the instructions in the Readme file with respect to installing, testing and removing (if necessary) the test fix.
When installing a test fix in a multi-partition database partition environment, the system must be offline and all computers participating in the instance must be upgraded to the same test fix level.

Results

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before you begin

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. For information about the types of maintenance contracts available, see "Premium Support" in the *Software Support Handbook* at: techsupport.services.ibm.com/guides/services.html

Procedure

Complete the following steps to contact IBM Software Support with a problem:

1. Define the problem, gather background information, and determine the severity of the problem. For help, see the "Contacting IBM" in the *Software Support Handbook*: techsupport.services.ibm.com/guides/beforecontacting.html
2. Gather diagnostic information.
3. Submit your problem to IBM Software Support in one of the following ways:
   - Online: Click the **ESR** (Electronic Service Request) link on the IBM Software Support, Open Service Request site: www.ibm.com/software/support/probsub.html
   - By phone: For the phone number to call in your country/region, go to the Contacts page of the *Software Support Handbook*: techsupport.services.ibm.com/guides/contacts.html

   Note: If you require support for any IBM product that is packaged as part of the Db2 pureScale software stack, then open a service request or problem management record (PMR) for the IBM Db2 pureScale Feature. Opening a PMR for the Db2 pureScale Feature helps resolve problems more efficiently.

What to do next

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Software Support website daily, so that other users who experience the same problem can benefit from the same resolution.

Exchanging information with IBM

To diagnose or identify a problem, you might need to provide IBM Support with data and information from your system. In other cases, IBM Support might provide you with tools or utilities to use for problem determination.

Sending information to IBM Support

You can submit data to IBM Software Support by FTP or by using the Electronic Service Request (ESR) tool. Instructions are provided here.

Before you begin

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

About this task

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

- FTP
- Electronic Service Request (ESR) tool

These and other data exchange methods that might be available are explained on the IBM Support website.

Procedure

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

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

What to do next

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

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

Receiving information from IBM Support

Occasionally an IBM technical-support representative might ask you to download diagnostic tools or other files. You can use FTP to download these files.

Before you begin

Ensure that your IBM technical-support representative provided you with the preferred server to use for downloading the files and the exact directory and file names to access.

Procedure

To download files from IBM Support:

1. Use FTP to connect to the site that your IBM technical-support representative provided and log in as `anonymous`. Use your email address as the password.
2. Change to the appropriate directory:
   a. Change to the `/fromibm` directory.
      ```
      cd fromibm
      ```
   b. Change to the directory that your IBM technical-support representative provided.
      ```
      cd nameofdirectory
      ```
3. Enable binary mode for your session.
   ```
   binary
   ```
4. Use the **`get`** command to download the file that your IBM technical-support representative specified.
   ```
   get filename.extension
   ```
5. End your FTP session.
   ```
   quit
   ```

Subscribing to Support updates

By subscribing to receive updates about Db2, you can receive important technical information and updates for specific IBM Support tools and resources.

About this task

You can subscribe to updates by using one of two approaches:

RSS feeds and social media subscriptions
The following RSS feeds and social media subscriptions are available for Db2:

- Db2 for Linux UNIX and Windows
- Db2 Merge Backup for Linux UNIX and Windows

For general information about RSS, including steps for getting started and a list of RSS-enabled IBM web pages, visit the IBM Software Support RSS feeds site.

My Notifications
With My Notifications, you can subscribe to Support updates for any IBM product. My Notifications replaces My Support, which is a similar tool that you might have used in the past. With My Notifications, you can specify that you want to receive daily or weekly email announcements. You can

specify what type of information you want to receive, such as publications, hints and tips, product flashes (also known as alerts), downloads, and drivers. My Notifications enables you to customize and categorize the products about which you want to be informed and the delivery methods that best suit your needs.

Procedure

To subscribe to Support updates:

1. Subscribe to the Db2 RSS feeds by pointing your browser to the URL for the one of the RSS feeds and clicking **Subscribe Now**.
2. Subscribe to My Notifications by going to the IBM Support Portal and click **My Notifications** in the **Notifications** portlet.
3. Sign in using your IBM ID and password, and click **Submit**.
4. Identify what and how you want to receive updates.
   a. Click the **Subscribe** tab.
   b. Select the appropriate software brand or type of hardware.
   c. Select one or more products by name and click **Continue**.
   d. Select your preferences for how to receive updates, whether by email, online in a designated folder, or as an RSS or Atom feed.
   e. Select the types of documentation updates that you want to receive, for example, new information about product downloads and discussion group comments.
   f. Click **Submit**.

Results

Until you modify your RSS feeds and My Notifications preferences, you receive notifications of updates that you have requested. You can modify your preferences when needed.

Index