



IBM DB2® Universal Database™
DB2 Problem Determination Tutorial Series

Introduction to Problem Determination

Table of Contents

Introduction to Problem Determination.....	4
About this tutorial.....	4
Tutorial objectives.....	4
Audience and assumptions.....	4
Pre-tutorial setup.....	4
Tutorial conventions used.....	4
About the authors.....	5
Problem description.....	6
Introduction.....	6
What is the problem?.....	6
My application failed!.....	6
Interpreting DB2 informational messages.....	7
What is the environment and where is the problem originating?.....	8
When does the problem happen?.....	8
Under which conditions does the problem happen?.....	9
Is the problem reproducible?.....	9
Conclusion.....	10
DB2 data collection.....	11
Introduction.....	11
DB2 files.....	11
db2diag.log file.....	11
db2diag.log example.....	12
Trap files.....	13
Dump files.....	14
Messages files.....	14
db2support.....	14
Conclusion.....	15
Other data collection.....	16
Introduction.....	16
Operating systems.....	16
Applications and third-party vendors.....	16
Hardware.....	17
Conclusion.....	17
Problem investigation.....	18
Introduction.....	18
System problems.....	18
Instance problems.....	18
Database problems.....	19
Utility problems.....	19
Transactional problems.....	20
Conclusion.....	20
Related and fixed problem Research.....	21

Introduction.....	21
DB2 Technical Support site	21
Another example situation.....	21
Finding related problems	22
What is an APAR?	23
Select APAR results.....	23
DB2 Support site accessing APARs	24
An APAR	24
Open vs closed APARs	25
Normal vs. HIPER APARs	26
What to do about HIPER APARs	27
Open APARs.....	27
APARs by FixPaks	27
Newsgroups.....	28
Conclusion	29
Summary.....	30
What you should have learned.....	30

Introduction to Problem Determination

About this tutorial

Tutorial objectives

This tutorial assists you in describing and investigating problems you may have with DB2, and isolating these problems using diagnostic information. Once you isolate a problem, you will be able to search known problems, identify circumvention, work-arounds, and availability of a fix. Also, by being familiar with finding known problems, you can avoid potential problems. These objectives are achieved by guiding you through examples of a few common types of problems.

Audience and assumptions

This tutorial applies to everyone who uses DB2 on Linux, OS/2, Windows® and UNIX® (AIX, Solaris, HP, etc). This includes users from DB2 Personal Edition to administrators of multiple partition DB2 EEE clusters. It is assumed that you are comfortable with the operating system you use and the tools that it provides; and you have working knowledge of DB2.

An excellent resource for general problem determination and the available problem support offerings is the *IBM Software Support Handbook*. It can be found at <http://techsupport.services.ibm.com/guides/handbook.html> . There are references to this document throughout this tutorial.

Some examples show specific command options which might change over time. All the commands and their options are shown in the DB2 documentation.

Pre-tutorial setup

DB2 installed, with a DB2 instance and a database created.

Tutorial conventions used

When a tool or utility is first mentioned it is shown in **bold** text.

All commands statements and their outputs are shown in a `monospaced` font.

About the authors

Lloyd D. Budd is a DB2 Support Analyst at the IBM Toronto Laboratory in Markham, Canada. He is an IBM Certified Solutions Expert - DB2 UDB Version 5, 6, & 7 Database Administration. Recently, he co-authored the redbook, *The Practical Guide to DB2 Data Replication Version 8*. His areas of expertise include logging, backup, and recovery.

Chris Fender is a technical manager on the DB2 Technical Support team. He has worked for IBM for 5 years. All his years working at IBM have been in the area of DB2 customer support specializing in the DB2 engine components, primarily in the fields of database recovery and data protection. He recently authored a PD/PSI chapter in the book *DB2: The Complete Reference*. He works at the IBM Toronto Laboratory, and is a recognized Professional Engineer in the province of Ontario.

You can reach Lloyd Budd and Chris Fender by locating their email address in the IBM Global Directory at <http://www.ibm.com/contact/employees/us> .

Problem description

Introduction

The first step in good problem analysis is to completely describe the problem. 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 is the problem?
- 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. This section provides several sample questions that you should ask, and an example from a common type of problem situation.

What is the problem?

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

- Who or what is reporting the problem?
- What are the symptoms?
- What is the business impact?

The best way to understand how and when you would ask these questions is through the use of an example.

My application failed!

This example, appropriately titled "My application failed!" is a common situation that every software developer or database administrator encounters at some point. Usually in this situation, a user rushes into a room, phones, or pages you screaming the phrase quite loudly.

Once the user calms down, the problem determination begins. You start by asking "What are the problem symptoms?" The alluded response of "My application failed" answers the question of the first layer that is reporting the problem (the application), and can be followed up with the question "What was the application doing?" In this example, assuming the application is coded well with its own logging, that question is answered by "the application was processing payroll information, and its logs show an SQL INSERT fail multiple times with a return value of SQLCODE SQL0289N."

You now have an vague overview of the problem, where it occurred, and an SQLCODE to begin investigating. To pursue further, you need to know what the message means.

Interpreting DB2 informational messages

You can start your investigation from the reported SQLCODE, as DB2 informational messages are always returned in the form of CCCnnnnnS. The **CCC** identifies the DB2 component returning the message, the **nnnnn** is a four or five digit error code, and the **S** is a severity indicator. The **SQL** component identifier in this example represents a message from the Database Manager. Here is the complete list for your reference:

- SQL Database Manager messages
- DB2 Command Line Processor messages
- ASN Replication messages
- CLI Call Level Interface messages
- SQJ Embedded SQLJ in Java messages
- SPM Synch Point Manager messages
- DBI Installation or configuration messages
- DBA Control Center and Database Administration Utility messages
- CCA Client Configuration Assistant messages
- DWC Data Warehouse Center messages
- FLG Information Catalog Manager messages
- SAT Satellite messages

You can completely identify any DB2 return code message using the DB2 command line by separating the **db2** command and the error code with a question mark:

```
D:\>db2 "? sql0289"

SQL0289N Unable to allocate new pages in table space
"<tablespace-name>".
```

Most error messages contain further information than what is shown here, including an explanation of the error and potential user responses (removed above to conserve space). You can find full information about the DB2 message format, and a listing of all messages in the DB2 UDB

Messages Reference, Volumes 1 & 2. The message text is a great quick reference for investigating any problem, and should always be reviewed when describing your problems.

Now that you know the user is encountering what looks to be a condition where a tablespace cannot allocate any new pages, further probing into the example will reveal other problem symptoms. Questions like "Is this the only application that is failing?" help here. This is an excellent start to describing the problem, but you still need to round out the complete picture.

What is the environment and where is the problem originating?

Determining where the problem originates is not always easy, but is one of the most important steps in resolving a problem. Many layers of technology almost always 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 application running local on the database server or on a remote server?
- Is there a gateway involved?
- Does the database reside on individual disks, or on a RAID 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 aren't meant to run together, or haven't been fully tested together.

For simplicity, our example involves a failing application running directly on the server in a supported configuration, so we will only need to concentrate on server-side investigation.

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?

- What sequence of events lead up to the time the problem is reported?
- Do any automatic scripts or cron jobs start at the same time as the problem?

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. The last question above is hinting at the next topic.

In the example, the user indicates that the problem is reproducible, so the exact time the problem occurred is not as significant as in a single incident occurrence, however determining the steps leading up to the failure is. In this case, simply running the standalone application is enough to reproduce the problem, so a detailed time line of events is not required.

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 happen only while something else is running?
- Does a certain sequence of events need to occur for the problem to surface?
- Do other applications fail at the same time?

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

In the example, as previously described, the problem can be reproduced standalone, so there don't seem to be any external conditions or dependencies on other applications to help explain the problem cause.

Is the problem reproducible?

From a problem description and investigation standpoint, the "ideal" problem is one that is reproducible. Almost always, with reproducible problems you have a larger set of tools or procedures available to use to help your investigation, consequently reproducible problems are 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 reoccurring. Some reproducible problems present considerable pressure and serious time constraints to resolve.)

- Can the problem be recreated on a test machine?
- Is the problem platform-specific, or common to multiple platforms?
- Are multiple users or applications encountering the same type of problem?

- Can the problem be recreated by running a single command, a set of commands, or a particular application?
- Can the problem be recreated by entering a command/query from a DB2 command line, or only using an application?

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.

Our example is reproducible. The impact appears to be limited to this one user; however she indicates that the same application completes fine in her test environment, so we are limited to perform problem determination in the production environment in which it was found. This means that you may be under considerable pressure to solve the problem, and have limited time. You must move fast!

Conclusion

Describing a problem accurately and completely may be easy for some problems, but very difficult for others. (The difficulty usually increases as the environmental complexity increases). However, the questions that you need to ask are usually the same: who, what, where, when, and how. For an excellent checklist of questions to ask, see the Problem Resolution Worksheet under the 'Before Contacting IBM' section of the *IBM Software Support Handbook* (<http://techsupport.services.ibm.com/guides/handbook.html>). It will help you to get a handle on the problem.

To summarize the description of the example for this section:

- An application is reporting an SQL0289N during an SQL INSERT
- The application fails when run local to the production database server
- The failure can be reproduced standalone

Now that you have good problem description skills, you need to develop your problem investigation and determination skills. The next section will concentrate on this area, and the available information to analyze most problems with DB2.

DB2 data collection

Introduction

DB2 (with each environment it runs in) provides information for you to solve problems you may encounter with it. This section will explain what information is available, and how you can find and use it.

DB2 files

The most important information you can use to investigate DB2 problems are files generated by DB2 itself. Some of these files include:

- db2diag.log file
- Trap files
- Dump files
- Messages files

These files are generated or updated when different events or problems occur. In this section of the tutorial, you will see how to review each type of file, and how to interpret the information.

db2diag.log file

The `db2diag.log` is the file most often used for DB2 problem investigation. You can find this file in the DB2 diagnostic directory, defined by the `DIAGPATH` variable in the Database Manager Configuration. (Note that in this section, you will see that many of DB2's diagnostic files will be created in this directory.) By default the directory is defined as follows:

- UNIX: `$HOME/sqllib/db2dump`
 - `$HOME` is the DB2 instance owner's home directory
 - For example, a default `db2diag.log` for an instance owner of `fender` on UNIX would exist in `/home/fender/sqllib/db2dump/db2diag.log`
- Windows or OS/2: `INSTALL PATH\SQLLIB\<DB2INSTANCE>`
 - `INSTALL PATH` is the directory where DB2 is installed
 - For example, a default `db2diag.log` for an instance owner of `DB2` on Windows would exist in `C:\Program Files\SQLLIB\DB2\db2diag.log`

The Database Manager Configuration also controls how much information is logged to the `db2diag.log` through the use of the diagnostic level, or `DIAGLEVEL` variable. Valid values can range from 0 to 4, with the following differences between each:

- 0 - No messages
- 1 - Severe error messages
- 2 - Only error messages
- 3 - All error and warning messages (default)
- 4 - All error, warning, informational, and internal diagnostic messages

The default diagnostic level of 3 is usually sufficient for problem determination. Setting it to 4 may cause performance issues due to the large amount of data recorded into the file. You should adjust the setting according to each problem you encounter, the amount of data you require to investigate, and the environment in which each problem occurs.

db2diag.log example

The `db2diag.log` file contains both administrative information for normal operations (for example the load utility), as well as diagnostic information for failures and abnormal operations (for example crash recovery). In DB2 UDB Version 8, the file is split into two separate files to help administrators and support personnel discriminate between these events. Here is an example of a failing command on the Windows platform along with its corresponding `db2diag.log` entry at a diagnostic level of 3 (you can recreate this problem on your own computer by replacing the `z:\` with a path that does not exist):

```
D:\> db2 backup db sample to z:
SQL2036N  The path for the file or device "z:\" is not valid.
```

```
2002-10-28-19.12.54.825000 Instance:DB2 Node:000
PID:1124(db2syscs.exe) TID:680 Appid:none
database_utilities sqluMCTestDevType4Backup Probe:60

Media controller -- invalid device path: z:\
```

You can clearly see from both the error message and the corresponding `db2diag.log` entry that the path `z:\` does not exist, which is the reason for the backup failure. There are further entries dumped out into the `db2diag.log` at diagnostic level 3, but this gives you a good idea of the type of messages created on a failing utility command.

Here is a breakdown of the entry in the `db2diag.log` above:

- The message was generated at **2002-10-28-19.12.54.825000**
- The instance name is **DB2**
- The partition number (or node, 0 for EE) is **0**
- The process name is **db2syscs.exe**
- The process ID (PID) is **1124**
- The thread ID (TID) is **680**

- There is no application ID (Appid)
- The DB2 internal component identifier is **database_utilities**
- The DB2 internal function identifier is **sqluMCTestDevType4Backup**
- The unique error identifier (probe ID) within the reported function is **60**

The last part of the message entry is a message which often includes error codes, page dumps, or other detailed information. Sometimes this information will be complex, but usually it will give you an idea of the type of operation that is causing the failure, along with some supporting information to help the investigation.

Trap files

Whenever a DB2 process receives a signal or exception (raised by the operating system as a result of a system event) that is recognized by the DB2 signal handler, a trap file is generated in the DB2 diagnostic directory. The files are created using the following naming convention:

- UNIX: *pppppp.nnn*
 - *pppppp* : the process ID (PID)
 - *nnn* : the node where the trap occurred
 - Eg. t123456.000
- Intel: *DBpppptt.TRP*
 - *ppp* : the process ID (PID)
 - *ttt* : the thread ID (TID)
 - Eg. DB123654.TRP

Depending on the signal received or the exception raised, the existence of these files can indicate different extremes of consequences. These consequences can range from the generation of a simple stack traceback for additional diagnostics, to a complete DB2 instance shutdown due to a serious internal or external problem. You can obtain a list of all available signals for your operating system from the following files:

- UNIX : /usr/include/sys/signal.h
- Windows (requires the software development kit): Winnt.h
- OS/2 (requires the software development kit): bsexcpt.h

Other tutorials in this series provide further information regarding trap file investigation, and how to generate them for problem diagnosis.

Dump files

When DB2 determines that internal information needs to be collected, it will often create dump files in the diagnostic path (as described earlier in this tutorial). These files are of two types and are generated with the following two formats:

Type 1: Binary dump files

- UNIX: *ppppppp.nnn*
 - *ppppppp*: the process ID (PID)
 - *nnn*: the node where the problem occurred
 - Eg. 123456.000
- Windows and OS/2: *pppttt.nnn*
 - *ppp*: the process ID (PID)
 - *ttt*: the thread ID (TID)
 - *nnn*: the node where the problem occurred
 - Eg. 123654.000

Type 2: Locklist dump files

- UNIX: *lppppppp.nnn*
 - *ppppppp*: the process ID (PID)
 - *nnn*: the node where the problem occurred
 - Eg. 1123456.000
- Windows and OS/2: *lpppttt.nnn*
 - *ppp*: the process ID (PID)
 - *ttt*: the thread ID (TID)
 - *nnn*: the node where the problem occurred
 - Eg. 1123654.000

Messages files

Some DB2 utilities like BIND, LOAD, EXPORT, and IMPORT provide an option to dump out a messages file to a user-defined location. These files contain useful information to report the progress, success or failure of the utility that was run. You should take advantage of generating these files to ensure you are obtaining the most possible information in case of a problem.

db2support

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

The db2support utility is designed to automatically collect all DB2 and system diagnostic information available (including information described in previous pages). It has an optional interactive "Question and Answer" session available to help collect information for problems that you may want additional assistance investigating. Using db2support avoids possible user errors as you don't need to manually type commands such as "GET DATABASE CONFIGURATION FOR <database name>" or "LIST TABLESPACES SHOW DETAIL". Also, you don't require instructions on what commands to run or what files to collect, which makes information gathering for problem determination quicker.

This command has been in the DB2 product on Linux, OS/2, Windows, and UNIX, since version 7 Fixpak 4, and is continually being enhanced. For example, in version 7 FixPak 7 the default output is now broken down into separate text files rather than the previous one large HTML file, making it easier to read and parse. Executing `db2support -h` brings up the complete list of possible options you can run the utility with. The following basic invocation is usually sufficient for collecting most of the information required to debug a problem (note that if the `-c` option is used the utility will establish a connection to the database)

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

If further information is required, you need to review which extra options could be used to help. The output is conveniently collected and stored in a compressed ZIP archive, `db2support.zip`, so it can be transferred and extracted easily on any system.

Conclusion

The information provided in this section tells you how to obtain diagnostic information for analyzing DB2 problems. Most times these files are all that is required to determine the root cause of your DB2 problem, but sometimes more information is required to complete your investigation. The next section will briefly identify other areas outside of DB2 you can look at when trying to solve problems in your system.

Other data collection

Introduction

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

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

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

Operating systems

Every operating system has its own set of diagnostic files to keep track of activity and failures. The most common (and usually most useful) is an error report or event log. Here is a list of how this information can be collected:

- AIX: the `/usr/bin/errpt -a` command
- Solaris: `/var/adm/messages*` files or the `/usr/bin/dmesg` command
- Linux: the `/var/log/messages*` files
- HP-UX: the `/var/adm/syslog/syslog.log` file or the `/usr/bin/dmesg` command
- Windows NT® /Windows 2000®: the system, security, and application event log files and the `windir\drwtsn32.log` file (where `windir` is the Windows install directory)
- OS/2: the `INSTALL_DRIVE\OS2\SYSTEM\RAS\LOG*.DAT` file (where `INSTALL_DRIVE` is the OS/2 install directory)

There are always more tracing and debug utilities for each operating system. Refer to your operating system documentation and support material to determine what further information is available. Further information on this topic is also covered in the separate tutorial on "Combining DB2 and OS Diagnostics".

Applications and third-party vendors

As described in section 2, each application should have its own logging and diagnostic files. These files will complement the DB2 set of information to provide you with a more accurate picture of potential problem areas.

Hardware

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

Conclusion

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

Problem investigation

Introduction

Almost every problem encountered in DB2 can be categorized in one of five discrete problem types:

1. System problems
2. Instance problems
3. Database problems
4. Utility problems
5. Transactional problems

If you are able to identify which type of problem you are encountering, you will be able to quickly understand what information is required to debug each type of problem, and how to begin your investigation into each.

System problems

When you encounter a problem that seems "larger than life", often it is. For example, if you cannot telnet to a DB2 server machine without receiving an error, or you cannot type simple directory listing commands without causing a hang, or you cannot perform a common file move command without forcing a machine system dump, then you probably have an entire system problem. These problems do happen, and when they do there is little you can do from a DB2 investigation standpoint. You would need to involve your operating system administrators to diagnose these types of problems, as DB2 (like any other software on the machine) is likely the victim of a larger issue.

Instance problems

If you receive a message indicating that the database manager needs to be started when the instance should already be active, it's possible you have hit an instance problem. This usually happens when a DB2 process receives a signal from the operating system which causes DB2 to invoke its signal handlers brings the instance down. Some examples of why these signals could be raised include:

- Some other software is overwriting DB2's stack
- An authorized user is manually issuing kill signals against DB2 processes
- DB2 is causing them itself (potentially due to a software defect)

When instance problems such as these occur, almost always there will be essential information contained in the db2diag.log. Also trap and dump files may have been generated. As soon as you have brought the database manager back online, you would start by looking at those files to determine the problem cause.

The “DB2 Problem Determination Tools” tutorial in this series provides a sample trap file and explains how the function at the top of the stack traceback is usually the culprit that’s causing the problem. Where DB2 is concerned, the function at the top of the traceback is an excellent term to use for researching your problem in the list of known DB2 defects. An example of a good search term, taken from the example in Section 3 of this tutorial, is `sqlumCTestDevType4Backup`. More details on researching known problems are outlined in later sections in this tutorial.

Database problems

If most users of the same database are experiencing a common problem, which could indicate that the problem is with the database itself. Examples could range from a performance problem to a database’s becoming inaccessible because of a corrupted data page. Depending on the nature of the reported problem, you would use different techniques for problem determination.

For debugging database performance issues, please refer to the "Performance Problem Determination" tutorial in this series.

For debugging database availability problems, concentrate on the `db2diag.log` and relevant system information (outlined in Section 4 of this tutorial). You can usually combine those in order to resolve most database level crash issues. (Note: often you will see trap or dump files created at the time of these occurrences, but usually that information is dumped to provide DB2 Service analysts with detailed information on what DB2 was doing internally at the time of the failure. It is not as useful to you for general investigation as the `db2diag.log` file.)

Utility problems

The example used in Section 3 of this tutorial is a good example of a utility problem. In that example, a DB2 backup command fails because an invalid target path was supplied. The `db2diag.log` is generally the best file to refer to when debugging these types of problems.

In addition, as explained in Section 3 of this tutorial, DB2 utilities have the option of creating their own message logs. (This is not available for the DB2 backup command.) These files are usually just as important as the `db2diag.log` for investigating problems, as they provide essential information pertaining only to that operation.

Using a combination of the `db2diag.log` file and utility message files is usually the best method for you to start your investigation into utility problems. Remember that if the utility is being run from a remote client, then you may have connectivity issues that come into play. If that is the case, then please refer to the tutorial on “Connectivity Problem Determination” for further information on how to debug that aspect of the problem. In the case of environments with multiple partitions, even locally run utilities have communication aspects that may come into play.

Transactional problems

A DB2 transactional problem can occur with any SQL command you run within DB2. This includes common SELECT, INSERT, UPDATE, and DELETE statements, along with any other commands listed in the *DB2 SQL Reference* manual. An SQL problem can usually be categorized as a performance issue, or a command failure.

For debugging SQL performance issues, please refer to the "Performance Problem Determination" tutorial in this series.

For debugging SQL failures, the returned message information is usually the best way to identify the problem and its cause, and to determine potential solutions. You saw in the example presented in Section 2 of this tutorial how these messages can be powerful tools in your problem determination toolbox.

Conclusion

As you can see, each problem you encounter using DB2 may require different approaches when diagnosing the problem. Once you have accurately described the problem you are encountering, have categorized it to be one of the general problem types explained in this section, and have collected all relevant information pertaining to that type of problem, you need to use all of your technical problem determination skills to obtain problem resolution.

Although not discussed in this tutorial, using the DB2 trace facility (`db2trc`) is an excellent way to debug run-time and reproducible problems. The DB2 trace facility is covered in other tutorials in this series.

You will only become comfortable with debugging each of these different types of problems through practice and experience. It is a good idea to generate some known problem yourself and use the techniques you have learned. It's better to debug a problem in a controlled, relaxed environment than to experience it first hand when you are trying to get your database back up and running in the middle of the night.

Now that you can describe, collect data for a problem, you need to know how to effectively search for known or related problems. This information is covered in the next sections of this tutorial.

Related and fixed problem Research

Introduction

Now that you have a good understanding about the elements of a problem, you can research similar problems, and learn about workarounds, and fixes. The first pages in this section discuss documentation and searching. The majority of the section helps you understand how DB2 publishes defects (APARs) externally. This section ends by looking at additional resources for researching DB2 problems.

DB2 Technical Support site

The first place you should look if you are experiencing a DB2 problem is the documentation, particularly the Release Notes. The DB2 Technical Support site for Linux, OS/2, Windows, and UNIX has the most recent copies of the documentation. It is located on the Web at <http://www.ibm.com/software/data/db2/udb/winos2unix/support>, and hosts this tutorial.

It is also the best place to start your search for related problems.

Another example situation

Here is another "My application failed!" problem like you saw in Section 2. It will help you practice your problem description and investigation skills, as well as lead into how you can research known issues.

User account of: DB2 is crashing!

DB2 keeps crashing when I try to access it with my application. It is near the end of my work day, and I am sure that there are only a few people using this database at this time. I am running the same jobs that I run during the day. This is work that I have to get done today! I am also very concerned about the stability of our environment!

By working with your user, and looking at the diagnostic data, you generate the following problem description:

- The environment AIX 4.3.3 / DB2 EE v7.2 fixpak 6.
- This instance is used as a gateway to a zSeries host.
- SNA is being used for communication.
- The DB2 instance is crashing.
- The crashing does not seem to be related to a particular utility or scheduled job.
- It occurs only when few users are using the gateway.
- This new problem is not reproducible, but has occurred more than once.

- There have been no changes to the hardware or software configurations for many months.
- There are no error messages on the host.
- The AIX `errpt` lists a number of software program abnormally terminated, and the program name is a DB2 executable.
- Looking in the `db2dump` directory, there are quite a few `tnnnnn.000` (trap) files. Opening these files you confirm that they are in plain text and are stack tracebacks. All but one was for signal 6 (SIGABRT), which is self-induced when DB2 is shutting down. At the top of the other trap file it shows a date and time close to the last occurrence, and it is signal 4 (SIGILL). Halfway into the file the stack is shown, the first line (the top of the stack) shows:

```
offset      178 in function
sqlccgetattr__FP15sqlcc_comhandleP10sqlcc_attrP10sqlcc_cond
```

The effect is bad, but the business impact is not critical as it does not occur at a busy processing times.

Finding related problems

Go to the DB2 Technical Support Web site (<http://www.ibm.com/software/data/db2/udb/winos2unix/support>) and work along through the scenario.

Search terms that are successful often involve:

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

In section 5, you learned about what a trap file is. Traps and stack tracebacks are covered in detail in another tutorial in this series. The UNIX traceback on the previous page shows the trapping function to be `sqlccgetattr`. Searching the DB2 documentation you found no results. Searching DB2 APARs will show results similar to the following picture:

Search results summary	
You searched for " sqlccgetattr "	
Search results were returned in the following categories:	
DB2 Authorized Program Analysis Reports (APARs)	Results
APAR Library	2

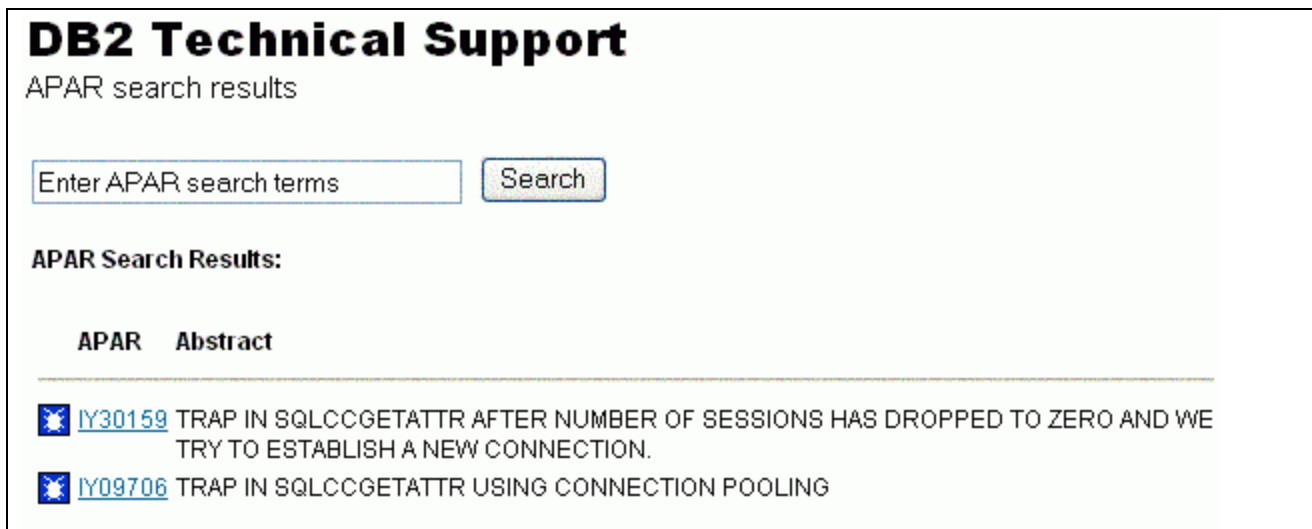
Before you click on that to see what documents it contains, you should learn what an APAR is.

What is an APAR?

IBM resolves defects discovered by customers in Authorized Program Analysis Reports (APARs). Seldom will you hear the term “Authorized Program Analysis Reports”, as the acronym is very common. An APAR is simply an externalized view of an IBM defect. APARs have unique identifiers. On the next page you will see two DB2 APAR identifiers, IY16397 and IY09706. DB2 APARs are always specific to a particular version, but may affect multiple products in the DB2 family running on multiple platforms.

Select APAR results

With your understanding of what APARs are, let’s look at the results of clicking on **APAR Library**, as shown in this picture:



The screenshot shows the 'DB2 Technical Support' APAR search results page. It features a search bar with the placeholder text 'Enter APAR search terms' and a 'Search' button. Below the search bar, the text 'APAR Search Results:' is displayed. A table with two columns, 'APAR' and 'Abstract', lists two results. The first result is for APAR IY30159, with the abstract 'TRAP IN SQLCCGETATTR AFTER NUMBER OF SESSIONS HAS DROPPED TO ZERO AND WE TRY TO ESTABLISH A NEW CONNECTION.' The second result is for APAR IY09706, with the abstract 'TRAP IN SQLCCGETATTR USING CONNECTION POOLING'.

APAR	Abstract
IY30159	TRAP IN SQLCCGETATTR AFTER NUMBER OF SESSIONS HAS DROPPED TO ZERO AND WE TRY TO ESTABLISH A NEW CONNECTION.
IY09706	TRAP IN SQLCCGETATTR USING CONNECTION POOLING

Two APARs are listed. Clicking an APAR identifier takes you to a listing of that APAR. What is different about these two APARs?

If you guessed that the wording is slightly different, then you are correct! More importantly, you would discover after following the links that the DB2 version that they are for is different as well. DB2 APAR IY16397 is for version 7, and IY09706 is for version 6.

If an APAR describes a problem you are having, but it is for a different version, by searching using terms in that APAR, you may find an APAR for your version. That would have been the case if you were only aware of IY09706, but were encountering the problem in version 7.

For the trap in `sqlccgetattr`, you now know a solution. Install FixPak 7 or a more recent FixPak on your DB2 server.

There are other aspects of APARs to learn about.

DB2 Support site accessing APARs

Each page on the DB2 Technical Support site has a navigation menu on the left. Clicking the **APARs** selection takes you to a page where you can:

- Search for all APAR that match your search string
- View HIPER APAR
- View open APAR
- View APAR by FixPak

The first option to search for DB2 APARs is the same as the search you just did. Each of the views shows the most recent APARs first; they are listed in reverse chronological order.

An APAR

Enter `IY16397` in the search string, and click Search. Click through the pages until you are viewing the text of this APAR.

DB2 APAR `IY16397` is for another backup problem, but here the problem was that DB2 was doing something incorrectly. The nature of this problem requires many more details than the last APAR. Here is the APAR text:


```
Abstract:
SQL2025N RC=-50 ON A DB2 BACKUP TO ADSM AFTER A NODE IS ADDED
THAT DOES NOT CONTAIN DATA (EG. TEMP TABLESPACE)

Status:
CLOSED - 2001-05-12

APAR First Reported In:
DB2 UDB Enterprise Extended Edition for AIX v710
Note: APARs are not necessarily limited to one specific product.

APAR Description:

ERROR DESCRIPTION:
On backup of a node that was added in v5 and migrated to v7 FP2,
it failed with an SQL2025N rc=-50 from ADSM.

LOCAL FIX:
Add data on the empty node.

PROBLEM SUMMARY:
The estimate size we send to TSM is 0, so the dsmSendObj TSM
api was not flushing its buffer, and we would fail on the
dsmSendData because the dsmSendObj did not get processed. We
need to give an estimate size larger than the TCP buffer size
to ensure the dsmSendObj gets flushed.

TEMPORARY FIX:

First Fixed In:
DB2 V7 FixPak 3

Note: With the exception of Fixpak 2a for V7, a letter appended to
the FixPak number indicates that it is an Interim FixPak. Interim
FixPaks are temporary solutions. The official DB2 FixPak must be
applied when it is available.

Latest DB2 FixPak containing a fix for this APAR:
DB2 V7 FixPak 7
```

Look at parts of the APAR. The abstract is a brief description of the defect. The *error description* describes in detail the symptoms experienced. Depending on the complexity of the problem, this information will allow you to confirm if you are encountering the defect described. Both the *local fix* and *temporary fix* describe circumvention (if available), and corrective steps to work around the defect (if available). The *problem summary* may go into some detail about what causes the problem.

For this APAR, no temporary fix is listed. Has this problem been fixed in a FixPak?

Open vs closed APARs

Looking at APAR IY16397 you see:

APAR First Reported In:

```
DB2 UDB Enterprise Extended Edition for AIX v710
```

Note: APARs are not necessarily limited to one specific product.

The information we are interested in here is the DB2 version number. This APAR applies to version 7 of DB2.

The status is:

CLOSED - 2001-05-12

A status of closed means that it is included in an official FixPak, and the resolution has been verified in the first FixPak it is included in. Looking later in the record we see when it was first fixed:

First Fixed In:
DB2 V7 FixPak 3

Then it shows:

Latest DB2 FixPak containing a fix for this APAR:
DB2 V7 FixPak 7

A status of open would mean the APAR is not yet available in a FixPak. All open DB2 APARs, once documented, are accessible on the DB2 Technical Support site.

Normal vs. HIPER APARs

The second option on the DB2 Technical Support APAR Web page is to view APAR that are High-Impact or PERvasive (HIPER). The *IBM Software Support Handbook* (<http://techsupport.services.ibm.com/guides/handbook.html>) describes a HIPER APAR as:

- Problems that cause the destruction and/or contamination of customer data
- Problems that cause the customer to re-IPL, reboot, recycle, or restart one or more systems or subsystems
- Problems that cause a major loss of function
- Problems that cause severe impact to system performance or throughput.

The IBM Support team for DB2 includes pervasive problems in this list, as they are significant problems that have been encountered by multiple customers. The DB2 Technical Support Web page says, "HIPER APARs are critical bugs of which all DB2 customers should be aware." All APARs are externalized as quickly as possible, but HIPER APARs are externalized with urgency – often the same day that circumvention is developed.

The record of a HIPER APAR is the same format as all other APARs. The way to identify them as HIPER is to view the HIPER APAR list provided on the support Web site.

What to do about HIPER APARs

You should review each HIPER APAR that is not resolved in the DB2 version and FixPak that you are running. This helps you assess the risk exposure of staying at a particular FixPak level. Reading the abstract of the less severe open APARs also helps you assess the risk.

Open APARs

An open APAR is one that is not yet available in a FixPak. The third option on the DB2 Technical Support APAR Web page is to view open APARs.

Open HIPER APARs will also be listed here. However, only the HIPER APAR page shows whether they are HIPER.

Open APARs are likely to be resolved in the next FixPak after the version that they are opened against. There are two exceptions to this: an APAR might be identified too late in a FixPak's development and testing cycle to be included, or an APAR might be closed to be fixed in a future release (closed as FIN). The explanation of a FIN APAR is:

[If a] defect is determined to be of lower impact which does not require an immediate, permanent fix, [IBM] may defer the fix for a future release. APARs will reflect deferred fixes with a closing code of "FIN "(Fixed If there is a Next release) to designate plans for inclusion in a future release.

IBM Software Support Handbook

<http://techsupport.services.ibm.com/guides/handbook.html>

APARs by FixPaks

The last option on the DB2 Technical Support APAR Web page is to view APARs by FixPaks. Fixpaks are how DB2 delivers fixes. A FixPak includes numerous APARs. Other products use the terms *service pack* or *update*. DB2 releases FixPaks about every three months for supported versions.

Fixpaks include the APARs from the previous FixPaks, but you cannot skip moving between *point releases* of DB2. For example, if you are running DB2 at the version 7.1 FixPak 2 level and you want to go to the version 7.2 FixPak 7 level, you must install FixPak 3 first (which is equivalent to version 7.2). This type of intermediate step is described in the FixPak readme.

You may find it easier to navigate the APAR list (in plain text and html) contained in the FTP FixPak download directory. Below is an example of retrieving an APAR list. As you can see from the path entered, the information retrieved is for the AIX 4.3.3 32-bit DB2 version 7 FixPak 7 and its supporting documentation.

```
C:\>ftp ftp.software.ibm.com
```

```
User (none): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: ldbudd@ca.ibm.com
ftp> cd /ps/products/db2/fixes/english-us/db2aixv7/FP7_U484480/
ftp> dir
total 559832
-rw-rw-r--  1 4975300  1      159293 Sep  7 21:14 APARLIST.TXT
-rw-rw-r--  1 4975300  1      573489 Sep  7 21:14 APARLIST.html
-rw-rw-r--  1 4975300  1 285769770 Sep  7 21:06 FP7_U484480.tar.Z
-rw-rw-r--  1 4975300  1      22540 Sep  7 21:01 FixpakReadme.txt
lrwxrwxrwx  1 4975300  1         65 Jun 19 16:55 Release.Notes
ftp> mget APARLIST.html
```

The APAR list shows the newest APARs at the top, and the APARs are grouped by the FixPak that first contained them. Here is a fictional APAR list (for brevity):

```
*****
Product Name : DB2 UDB V7 32-bit for AIX
PTF Number   : U111111
Package Date : Fri Sep  6 12:34:43 EDT 2002
Build Level  : s020616
*****

APAR fixes included in U111111 (Fixpak 2)

APAR No.  Abstract
-----
GG04246   DATA LOSS WITH DATAPROPAGATOR APPLY WHEN THE SET HAS A LARGE
          NUMBER OF MEMBERS AND THE SOURCE IS A CCD.
GG04250   APPLY DOES NOT TERMINATE WHEN COPYONCE IS SPECIFIED AND
          THE SET FAILS WITH A -1.
GG04251   APPLY COULD MISS THE HINT AND RESULT IN MISSING ROWS IF
          CAPTURE ENCOUNTERS -911

APAR fixes included in U481406 (Fixpak 1)

APAR No.  Abstract
-----
GG04218   FULL REFRESH APPLY TROUBLE WHEN HANDLING MANY COLUMNS
GG04238   ASNLOAD SHOULD NOT ISSUE RUNSTAT
IC32474   CANNOT CREATE MAPPING FOR SYBASE TEXT DATA TYPE
```

The only difference between APARLIST.TXT and APARLIST.html, is the HTML allows you to click on an APAR identifier to go to the page that describes that APAR.

This finishes the discussion on APARs.

Newsgroups

Now, that you have a good understanding of APARs and their available resources, you can now apply the knowledge gained from the previous sections to solve all of your problems!

There is one great resource to remember – other DB2 experts like yourself!

Accessing newsgroups such as `comp.databases.ibm-db2`, you have access to the knowledge, problems, and solutions of others, on a massive number of topics. There is a very active DB2 community online that interact via newsgroups.

There are also IBM hosted DB2 newsgroups on the NNTP server `news.software.ibm.com`. There are quite a few, but `ibm.software.db2.udb` is the most heavily used.

There are a large number of other user oriented Web sites, but the USENET newsgroup will likely continue to be the most content rich.

Other resources that can assist problem determination are:

- DB2 Alerts email
(<http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/db2alert.d2w/report>)
- DB2 Support News
(<http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/newsletter.d2w/report>)
- Recent versions of the DB2 documentation
- DB2 Support Newsletter (<http://www-3.ibm.com/software/mailing-lists/>)
- Articles in DB2 Magazine (<http://www.db2mag.com/>)

Conclusion

You learned a large number of concepts related to finding related problems, including:

- Where the DB2 Technical Support site is on the Web
- Most of the content of the DB2 Technical Support site
- What an APAR is
- The properties of an APAR
- What the status of an APAR means
- Navigating the DB2 Technical Support site for APARs
- Searching newsgroups

Summary

What you should have learned

You now know how to identify and diagnose DB2 problems. You have a basic understanding of the diagnostic tools and files. You are familiar with common classes of problems and can find similar problems that others have encountered.

These skills are likely as important as your deep technical skills. You now have a solid basis for further development of your DB2 problem solving skills.