IBM DB2® Universal Database™
DB2 Problem Determination Tutorial Series

Problem Determination in a Multi Node Environment

Table of Contents

Problem Determination in a Multi Node Environment

Introduction

What this tutorial is about

This tutorial teaches you how to troubleshoot problems in a multi-node DB2 environment.

In this tutorial, you will learn:

1. How to narrow down which nodes in a multi-node environment are involved in a problem
2. How to understand problems in inter-node communication
3. How to diagnose problems with loading data into multiple nodes
4. How to understand problems with adding or removing nodes and redistributing data
5. How to understand DB2 EEE recovery problems

The primary audience for this tutorial is DB2 EEE database administrators (DBAs) and DB2 support engineers or consultants.

Pre-tutorial setup

The examples in this tutorial are specific to DB2 EEE running on the AIX operating system due to the assumption that this environment would be common to a larger group of DB2 EEE users. However, the concepts and information provided are relevant to DB2 EEE running on any platform.

In order to work through the examples in this tutorial, you should have done the following things:

- Installed DB2 EEE on two AIX machines (two partitions per physical node)
  The examples in this tutorial use two machines with hostnames **af01n002** and **af01n003** and with netnames **a_sw_002** and **a_sw_003**.
- Created a DB2 instance across four partitions with the /etc/services and sqllib/db2nodes.cfg definitions similar to the following:

/etc/services

```
xdb2inst1          26174/tcp
xdb2inst1_int      26175/tcp
DB2_db2inst1       26176/tcp
DB2_db2inst1_END   26177/tcp
```

db2nodes.cfg

Partition	Hostname	Port	Netname
0	af01n002	0	a_sw_002
1	af01n002	1	a_sw_002
2	af01n003	0	a_sw_003
3	af01n003	1	a_sw_003

The examples in this tutorial use the instance ID **db2inst1**.

- Executed the following commands and SQL statements to create the sample database and several database objects, and to generate some data files:

```
% db2sampl /database
(where /database is a path defined in a local file system on each
physical machine)

% db2 connect to sample
% db2 -tvf mytemp.ddl
```

mytemp.ddl

```
create temporary tablespace mytemp
managed by system
using ('/database/db2inst1/temp0') on node (0)
using ('/database/db2inst1/temp1') on node (1)
using ('/database/db2inst1/temp0') on node (2)
using ('/database/db2inst1/temp1') on node (3);
```

```
% db2 "export to staff.ixf of ixf select * from staff"

% db2 "export to staff.del of del select * from staff"
Edited the staff.del data file with around 15,000 records by
replicating from the existing output records.
```


About the author

Xiaomei Wang, B.C.S, M.Sc (Comp Sci), is a DB2 Certified Advanced Technical Expert and a senior member of the DB2 Support Team at the IBM Toronto Laboratory. She handles critical DB2 customer situations worldwide, using her expertise in the DB2 UDB Engine in non-partitioned (Enterprise, Workgroup, and Personal Editions) and partitioned (DB2 Enterprise- Extended Edition) database environments. She is also involved in trouble shooting education activities across IBM DB2 support centers worldwide.

You can reach Xiaomei Wang by locating her email address in the *IBM Global Directory* at http://www.ibm.com/contact/employees/us .

Narrowing down which nodes in a multi-node environment are involved in a problem

Start with a good problem description for troubleshooting in the DB2 EEE environment

A good problem description is essential in order to understand what is happening on your system. Once you understand the problem and the circumstances that led up to it, you can try to identify the source of the problem.

A good problem description always starts with the following things:

1. All error codes and error conditions
2. The reason code, if applicable
3. The actions that preceded the error
4. A reproducible scenario if possible
5. Identification of the partition where the error occurred or the failed operation involved

Here are examples of good and bad problem descriptions:

- I encounter an error every time I run my autoloader job. (Not good.)
- I'm getting a SQL6555 error every time my autoloader job reaches its split phase. The autoloader job was invoked on partition 0, and the splitting partition was defined on partition 3. (Good.)

What information should I collect?

Before you collect the diagnostic data, you need to understand if you have a single location, or multiple locations (paths) for the DIAGPATH directory where most of the diagnostic files reside based on the DBM configuration parameter. If you have changed the default DIAGPATH, then you may have separate paths on each physical node.

The following data is usually required for troubleshooting problems in the DB2 EEE environment:

- The SQL code plus any reason codes or the system error code
- A complete problem description, as explained previously
- Database code level (output from the **db2level** command)
- A copy of the db2nodes.cfg file
- Database manager configuration
- The database configuration: either provide the common configuration or the configuration for the specific partitions having the problem
- db2diag.log: if DIAGPATH is defined locally to each physical node, provide the db2diag.log files from the specific partitions having the problem

- The time of the error
- Any dump file listed in the db2diag.log file
- Any trap file in the DIAGPATH

In some cases additional data may be required in order to solve the problem.

How to narrow down problematic nodes in the DB2 EEE environment

In the DB2 EEE environment, the way to identify the partitions that are involved in a problem varies from problem to problem:

- Unexpected messages or SQL codes
  In a partitioned database environment, the database partition where an application is submitted is the *coordinator node*. Whenever a DB2 operation fails, the error message is returned to the coordinator node. Then, based on the nature of the operation and diagnostic data collected from the coordinator node, you can identify the rest of the nodes involved in this failed operation so that you can gather further diagnostic data from them.

- Abends (Abnormal ending)
  In a partitioned database environment, a DB2 instance could abend on all nodes or on some of the nodes. If you are not sure which nodes were shut down abnormally, check the DIAGPATH directories to see whether any trap files were generated at the specific time of the problem and where the trap files reveal the node numbers of partitions where the abend occurred. Or you may look through the db2diag.log files to search for the related *EDUCodeTrapHandler* entries, which are written into db2diag.log whenever there is an instance crash incident. The db2diag.log entry shows the node number information as well.

- Database or data corruption
  Normally the first time you notice a database corruption problem is when you fail to connect to a certain database partition or fail to access a certain database object. You can immediately conclude that the partition (or partitions) where the database object resides probably has the corruption. If you are able to schedule a long enough window, you may further use the **db2dart** utility to inspect all or part of the partitions and accurately identify the corrupted partitions.

- Loops and hangs
  When a specific operation or set of operations hangs, you might take a few application snapshots one minute apart on each partition, in order to determine what the status of the application is and whether any work is really being done. If the application status is Executing but no counters are increasing, then that's the partition you need to focus on.

- Slow performance
  For an operation with slow performance, you might again take a few application snapshots one minute apart on each partition, in order to determine what the status of the application is and how the work is being done. If the status is Executing and counters like rows-read or written are increasing, you might be able to evaluate the performance based on how fast the counters increase. Some DB2 utilities provide query commands, such as `load query` and `rollforward query`, which you can use to identify the partitions where performance is slower than you expect.

Understanding problems in inter-node communication

Problem starting a DB2 instance in a multi-node environment - .rhosts file consideration

Problem: db2start fails with SQL6048 communication error

Assume that you haven't edited the .rhosts file for your instance yet and that you see the following output from db2start at the console:

```
%db2start
11-10-2002 18:47:26     0   0
SQL6048N  A communication error occurred during START or STOP DATABASE
MANAGER processing.
11-10-2002 18:47:27     1   0
SQL6048N  A communication error occurred during START or STOP DATABASE
MANAGER processing.
11-10-2002 18:47:29     2   0
SQL6048N  A communication error occurred during START or STOP DATABASE
MANAGER processing.
11-10-2002 18:47:30     3   0
SQL6048N  A communication error occurred during START or STOP DATABASE
MANAGER processing.
SQL1032N  No start database manager command was issued.  SQLSTATE=57019
```

You can confirm the error in the db2diag.log file:

```
2002-11-10-19.26.49.566462   Instance:db2inst1   Node:000
PID:43844(db2start)   Appid:none
oper_system_services  sqloPdbExecuteRemoteCmd   Probe:50

6166 3031 6e30 3032 2e74 6f72 6f6c 6162       af01n002.torolab
2e69 626d 2e63 6f6d                           .ibm.com

2002-11-10-19.26.49.662979   Instance:db2inst1   Node:000
PID:43844(db2start)   Appid:none
oper_system_services  sqloPdbExecuteRemoteCmd   Probe:50

7273 6864 3a20 3038 3236 2d38 3133 2050       rshd: 0826-813 P
6572 6d69 7373 696f 6e20 6973 2064 656e       ermission is den
6965 642e 0a                                         ied..
…
```

Since DB2 EEE uses the **rsh** command to execute some commands such as db2start on all of the partitions, each partition must have the authority to perform remote commands on the other partitions. This can be done by updating the .rhosts file in the instance home directory. Since the instance home directory is NFS mounted, only one copy of the .rhosts file is necessary.

The db2diag.log file shown above indicates that permission is denied during the rsh operation. Next you would check the .rhosts file, and find that you haven't edited it yet. There are two ways to allow access to the instance ID by editing the .rshosts file:

Method 1:

```
        Hostname           Userid

 af01n002.torolab.ibm.com   db2inst1
 af01n002.torolab.ibm.com   db2inst1
 af01n003.torolab.ibm.com   db2inst1
 af01n003.torolab.ibm.com   db2inst1
```

Method 2:

```
            +              db2inst1
```

The first method is much more secure since it specifies the host name as well as the user ID that is granted access to the instance ID. The second method is not secure at all, since it allows any user ID of db2inst1 to access the instance ID, regardless of the host it belongs to. Method 2 should be used only in a lab environment.

After you edit the .rhosts file properly, you run db2start again. It should now be successful on all partitions.

Enable Fast Communication Manager (FCM)

Problem: db2start fails with SQL6031

You might simulate the SQL6031 error with the following steps:

Stop the DB2 instance by issuing the **db2stop** command:

```
% db2stop
```


Change your sqllib/db2nodes.cfg file by adding one more partition on the first physical node. For example, use the vi editor command to change the db2nodes.cfg file similar to the following one, while you can use any editor command you are comfortable with.

```
% vi db2nodes.cfg
```

```
  Partition   Hostname    Port    Netname

    0      af01n002     0      a_sw_002
    1      af01n002     1      a_sw_002
    2      af01n002     2      a_sw_002
    3      af01n003     0      a_sw_003
    4      af01n003     1      a_sw_003
```

In this example, the next attempt to issue the db2start command fails with the following message:

```
%db2start
SQL6031N  Error in the db2nodes.cfg file at line number "3".
Reason code "12".
```

To find out what reason code 12 refers to, execute the `db2 ?` command:

```
% db2 ? SQL6031
…
(12) The port value at line "<line>" of the db2nodes.cfg file in
the sqllib directory is not in the valid port range defined for
your DB2 instance id in the services file (/etc/services on
UNIX&reg;-based  systems).
…
```

In a partitioned database environment, most communication between database partitions is handled by the Fast Communications Manager (FCM). To enable the FCM at a database partition and allow communication with other database partitions, you must create a service entry in the /etc/services file. The FCM uses the specified port to communicate. If you have defined multiple partitions on the same host, you must define a range of ports.

If the /etc/services file is shared, you must ensure that the number of ports allocated in the file is either greater than or equal to the largest number of multiple database partitions in the instance. When allocating ports, also ensure that you account for any processor that can be used as a backup.

If the /etc/services file is not shared, the same considerations apply, with one additional consideration: you must ensure that the entries defined for the DB2 instance are the same in all /etc/services files (though other entries that do not apply to your partitioned database do not have to be the same).

The name of the service of the first port for FCM must be **DB2_instance-name**, and the name of the service of the last port must be **DB2_instance-name_END**.

Now you may verify the port definition in /etc/services:

```
% more /etc/services | grep db2inst1
xdb2inst1                   26174/tcp
xdb2inst1_int               26175/tcp
DB2_db2inst1                26176/tcp
DB2_db2inst1_END            26177/tcp
```

You can see that the number of ports defined for FCM in the /etc/services file is two, which is 26176 and 26177. However, the number of partitions defined on the physical node af01n002 in the sqllib/db2nodes file is three, which exceeds what is allocated in the /etc/services file.

You need to redefine the port range in /etc/services:

```
% vi /etc/services
DB2_db2inst1          26176/tcp
DB2_db2inst1_END      26178/tcp
```

And now db2start should be successful on all partitions.

Not enough FCM_NUM_RQB

FCM request blocks (RQBs) are the media through which information is passed between the FCM daemon and an agent, or between agents. The requirement for request blocks will vary according to the number of users on the system, the number of database partition servers in the system, and the complexity of queries that are run. When the system runs out of FCM request blocks, DB2 returns SQL6043, and dumps internal messages with the error code -6043 or 0xFFFFDC29 into the db2diag.log file.

The db2diag.log error messages may look like this:

```
2002-01-14-09.50.38.080676   Instance:db2inst1   Node:000
PID:65658(db2agent)   Appid:none
fast_comm_manager  sqlkf_alloc_rqb   Probe:30

DiagData
4e6f 2046 434d 2072 6571 7565 7374 2062       No FCM request b
6c6f 636b 2069 7320 6176 6169 6c61 626c       lock is availabl
652e 00                                       e..
..
2002-01-14-09.50.38.264865   Instance:db2inst1   Node:000
PID:79134(db2tcpcm)   Appid:none
```



```
base_sys_utilities  sqleGetAgent   Probe:75
.
Agent not allocated, sqlcode = -6043
```

You then need to increase the FCM_NUM_RQB dbm parameter to fix the problem:

```
% db2 update dbm cfg using FCM_NUM_RQB XXX
```
where *XXX* is the new value for the FCM_NUM_RQB parameter.

If not enough request blocks have been configured to support maximum connections as per the MAX_COORAGENTS dbm parameter, the `update dbm cfg` command returns this error:

```
SQL5153N  The update cannot be completed because the following
relationship would be violated:
"fcm_num_rqb >= max_coordagents * 2.5".
```

It is a good practice to perform database manager snapshot monitoring to check for the usage of FCM request blocks in the system so you can determine the best value for FCM_NUM_RQB. For example,

```
% db2 get snapshot for dbm
…
Node FCM information corresponds to            = 0
Free FCM buffers                               = 4093
Free FCM buffers low water mark                = 4085
Free FCM message anchors                       = 1534

Free FCM message anchors low water mark        = 1533
Free FCM connection entries                    = 1536
Free FCM connection entries low water mark     = 1531
Free FCM request blocks                        = 2012
Free FCM request blocks low water mark         = 2007
Number of FCM nodes                            = 4
```

Not enough FCM_NUM_BUFFERS

FCM buffers are 4 KB buffers that are used for internal communications (messages) both among and within the database servers in a partitioned database environment. They are not deallocated on every use. When a process has finished using them, they are returned to a "free" list for reuse. They are allocated all at once during db2start and deallocated all at once at db2stop time. During db2start, DB2 makes a single allocation of the entire FCM buffer pool, and then divides it into 4K chunks.

When the system runs out of FCM buffers, DB2 returns SQL6040, and dumps internal messages with the error code -6040 or 0xFFFFDC29 into the db2diag.log file.

You then need to increase the FCM_NUM_BUFFERS dbm parameter to work around the problem:

```
% db2 update dbm cfg using FCM_NUM_BUFFERS XXX
```
where *XXX* is the new value for the FCM_NUM_BUFFERS parameter.

However, in some cases even though there are free FCM buffers available at the time, certain DB2 operations may still fail with `SQL6040: No FCM buffers are available`.

For example,
you may see the following error messages in the db2diag.log file:

```
2002-09-30-22.28.16.453808   Instance:db2inst1   Node:002
PID:226042(db2agntp (SAMPLE) 2)   Appid:C0A86503.BEF6.020930031843
buffer_Q_services  sqlkqget   Probe:20   Database:SAMPLE
DIA9999E An internal error occurred. Report the following error code:
"alloc_buffer".

2002-09-30-22.28.16.464657   Instance:db2inst1   Node:002
PID:226042(db2agntp (SAMPLE) 2)  Appid:C0A86503.BEF6.020930031843
table_Q_services  sqlktrcv   Probe:50   Database:SAMPLE
DIA9999E An internal error occurred. Report the following error code :
"Line=00441, rc1=0xFFFFDC27, rc2=0xFFFFFFFF, rc3=0, msg=sqlkqrcv(conn)".
Dump File:/home/db2inst1/sqllib/db2dump/226042.002 Data:TAOB
…
```

In this case, the first thing you would do is to take a couple of snapshots:

```
% db2 get snapshot for dbm
Snapshot timestamp                = 09/30/2002 22:27:49.522724
Free FCM buffers                                = 1567
Free FCM buffers low water mark                 = 791
Free FCM message anchors                        = 5950
Free FCM message anchors low water mark         = 5757
Free FCM connection entries                     = 5617
Free FCM connection entries low water mark      = 4547
Free FCM request blocks                         = 6276
Free FCM request blocks low water mark          = 5215
Number of FCM nodes                             = 4

% db2 get snapshot for dbm
Snapshot timestamp              = 09/30/2002 22:28:49.284912
Free FCM buffers                                = 1340
Free FCM buffers low water mark                 = 791
Free FCM message anchors                        = 5862
Free FCM message anchors low water mark         = 5757
Free FCM connection entries                     = 5145
Free FCM connection entries low water mark      = 4547
Free FCM request blocks                         = 5805
Free FCM request blocks low water mark          = 5215
Number of FCM nodes                             = 4
```


The database manager snapshots between 22:27 and 22:28 show that low water mark for Free FCM buffer stays at 791, and the Free FCM buffer changes from 1467 to 1340. During that time interval, the error 0xFFFFDC27 is dumped in the db2diag.log file.

The current DB2 design is that DB2 would return no FCM buffer error to a low-priority requester when the low water mark for the free FCM buffer is less than 20% of the FCM_NUM_BUFFERS parameter. At 20% of FCM_NUM_BUFFERS DB2 only satisfies requests that are with medium or high priority. At 10%, only high priority requests are given FCM buffers.

Now you might issue the following get dbm cfg command to verify the value of the FCM_NUM_BUFFERS parameter:

```
% db2 get dbm cfg | grep FCM_NUM_BUFFERS
No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = 4096
```

In this case FCM_NUM_BUFFERS is defined as 4096. The dbm snapshots show that the low water mark for free FCM buffer is 791, which is less than 20% of FCM_NUM_BUFFER, where 4096(FCM_NUM_BUFFERS) * 20% = 819.2. At that moment a DB2 operation with the low priority request comes in, and can't be given a FCM buffer even though there are still FCM buffers available.

The requester's priority is assigned on the basis of the internal request types that agents want to communicate to other agents within the same node or to remote nodes. DB2 assigns it based on its impact on the integrity of the database and the instance. For example, in this case it might be that an SQL statement from a client translates into multiple FETCH requests in the engine. A FETCH request is treated as lower priority than, let's say, a unit-of-work ROLLBACK. Therefore the FETCH can only ask for "low" priority FCM buffers while the ROLLBACK has access to the remaining 791 FCM buffers as it is considered "high" priority.

Diagnosing problems with loading data into multiple nodes

Import utility - slow performance

The **import** command utilizes SQL INSERT statements to load data from an input file into a table or view. Each row is individually hashed based on the partitioning map and loaded to the appropriate partition. By default rows are inserted one at a time and the target partition responds to the coordinator partition with an SQL return code. During the import operation, all constraints are validated, all rows are logged, and triggers are fired if there is any. So the import utility is significantly slower than the load utility on large amounts of data.

To improve import performance, you can enable buffered inserts or use compound SQL as follows:

- Bind the import utility with the option (`Insert Buf`) to enable Buffered Insert

  Use the DB2 bind utility to request inserts to be buffered for better performance. The import package, db2uimpm.bnd, must be rebound against the database using the INSERT BUF option. For example:

  ```
  db2 connect to sample
  db2 bind db2uimpm.bnd insert buf
  db2 import from myfile of ixf insert into mytable
  ```

- Import the data with the Compound option
  Specify COMPOUND=n in the MODIFIED BY clause to group and send the specified number of insert SQL statements together. This limits the network traffic as compared to doing individual insert statements. The COMPOUND value may range from 1 to 100. For example:

  ```
  db2 connect to sample
  db2 import from myfile of ixf modified by compound=100 insert into
  mytable
  ```

You can practice the above suggestions by running the following two test scenarios:

Test 1:

Edit a shell script file called slow_import.sh:
slow_import.sh

```
db2 connect to sample
date
db2 import from staff.del of del replace into staff
date
```

Execute the slow_import.sh script file:

```
% slow_import.sh
…
Mon Nov 18 14:42:50 EST 2002
…
Mon Nov 18 14:43:12 EST 2002
```

Test 2:

Edit a shell script file called fast_import.sh:
fast_import.sh

```
db2 connect to sample
db2 bind /home/db2inst1/sqllib/bnd/db2uimpm.bnd insert buff
date
db2 import from staff.del of del modified by compound=100 replace
into staff
date
```

Execute the fast_import.sh script file:

```
% fast_import.sh
…
Mon Nov 18 15:05:41 EST 2002
…
Mon Nov 18 15:05:46 EST 2002
```

The above tests show that the default import operation takes 22 seconds, while the import with Buffered Inserts and the Compound option only takes 5 seconds to finish the operation. So Buffered Inserts and the Compound option improve the import performance significantly, which is important in the DB2 EEE environment where most of the imported data files are expected to be large.

Import utility - log full

By default an import operation is one unit of work (transaction). It doesn't commit until all the rows are loaded into the target partitions. When huge amounts of data are imported into a multi-node environment, the log space could easily be full by logging all the inserted rows, causing the import operation to fail with a log-full error, sql0964.

You might simulate the log full error with the following steps:

Decrease the log space size in order to easily simulate the problem situation:

```
% db2_all "db2 update db cfg for sample using LOGFILSIZ 4"
% db2_all "db2 update db cfg for sample using LOGPRIMARY 2"
% db2_all "db2 update db cfg for sample using LOGSECOND 0"
```

Recycle the database in order to have the db cfg update take effect:

```
% db2 force applications all
% db2 connect to sample
```

Import from the staff.del data file which contains around 15,000 records, and then you might receive the SQL0964 error:

```
% db2 import from staff.del of del insert into staff
SQL3109N  The utility is beginning to load data from file "staff.del".
SQL3306N  An SQL error "-964" occurred while inserting a row into the
table.
SQL0964C  The transaction log for the database is full.  SQLSTATE=57011
SQL3110N  The utility has completed processing.  "132" rows were read
from the input file.
```

You can confirm the error in the db2diag.log file:

```
2002-11-17-17.01.20.484111   Instance:db2inst1   Node:000
PID:112028(db2agent (SAMPLE) 0)   Appid:*LOCAL.db2inst1.021117185437
data_protection  sqlpgrsp   Probe:50   Database:SAMPLE

Log Full -- active log held by appl. handle 46
End this application by COMMIT, ROLLBACK or FORCE APPLICATION.

2002-11-17-17.01.20.559187   Instance:db2inst1   Node:000
PID:112028(db2agent (SAMPLE) 0)   Appid:*LOCAL.db2inst1.021117185437
data_protection  sqlpWriteLR   Probe:80   Database:SAMPLE
DIA3609C Log file was full.

ZRC=FFFFD509
```

To get around this you can either increase the amount of available log space or issue commits more frequently.

Method 1:
From the db2diag.log file, you can tell that partition 0 hits the log full error. You may increase the log space size on partition 0. You can either increase the log file size or the number of the log files. The following example is to increase the log file size on partition 0:

Switch to partition 0:
```
% db2 terminate
% export DB2NODE=0
```

Update the LOGFILSIZ db cfg parameter to increase the log space size:
```
% db2 update db cfg for sample using LOGFILSIZ 1000
```

Recycle the database to have the db cfg update take effect:
```
% db2 force applications all
% db2 connect to sample
```

Retry the import command:
```
% db2 import from staff.del of del insert into staff
```

Method 2:
Since this is an import with the insert option, a commit is only called once when all rows in the data file have been read and inserted into the table. If the COMMITCOUNT option on the IMPORT is used then this will reduce the number of records processed before a commit. Therefore, you may try using the COMMITCOUNT option as this should also help.

As an example, perform a commit after every 10 records are imported:

```
% db2 import from staff.del of del commitcount 10 insert into staff
```

A final option that you may consider is to use the LOAD command instead as its logging is minimal. Of course if your table is across multiple partitions then you must use the DB2 EEE autoloader.

Autoloader diagnostic files - what and where they are

The **autoloader** application uses the LOGFILE parameter defined in its configuration file as a base name to create the following autoloader log files:

- **<logfile>.split.cfg**
  Configuration file for all splitters
- **<logfile>.split.<3-digit-node-number>.log**
  Log file for each splitting partition that contains messages from the split process
- **<logfile>.load.<3-digit-node-number>**
  Message file for each loading partition that contains messages from the load processes if there is no message file specified in the load command

You may include a path in the LOGFILE parameter, however you must ensure the existence and accessibility of the path. The default value of the LOGFILE parameter is `./autoloader.log`. Keep in mind that in the case where there are multiple concurrent autoloader sessions, you must ensure either the base name of LOGFILE or the path name of LOGFILE is unique.

If the load command defines the MESSAGES option, DB2 generates the message files with the name like **<message file name>.<3-digit-node-number>** which is the message file for each loading partition that contains messages from the load processes. This message file only exists when the load command is issued with the MESSAGES <message file name> option. In this case <logfile>.load.<3-digit-node-number> is NOT created.

On each loading partition (also known as output partitions), as with the load operation in the DB2 EE environment, DB2 generates the load temporary files in a subdirectory called load/DB2*XXXXX*.PID/DB2*YYYYY*.PID under the database directory, where *XXXXX* and *YYYYY* are the pool (tablespace) ID and the object ID of the table involved in the load. This subdirectory only exists while the load is running and is deleted when the load finishes successfully. The users are warned not to tamper with this directory in any way. If they do, they can cause problems.

This directory typically contains the location file, the minimum recovery file, the messages file, and the load control files. The location file and the minimum recovery file should always be in this directory. If the user specifies the TEMPFILES PATH parameter in the db2 load command then the other files may be in an alternate directory.

- **Load<3-digit-node-number>.loc**
  The location file contains the directory where the control files and message files normally reside. It is controlled by the TEMPFILES PATH parameter to the load command. It defaults to the same directory as the location file. It is an ASCII file.
- **load<3-digit-node-number>.min**
  The minimum recovery file is used for LOAD REPLACE, just in case the control files go missing. For example, the user may specify the TEMPFILES PATH to point to some other directory and then delete that directory. LOAD REPLACE is still possible in this scenario. This is a binary file.
- **load<3-digit-node-number>.CT1, load<3-digit-node-number>.CT2**
  These are the load control files. Without these files a user cannot do a LOAD TERMINATE or RESTART. The files use a shadowing protocol to make sure that at least one version is correct at any time. They are binary files.
- **Load<3-digit-node-number>.msg**
  This is the binary form of the load messages file that appears at the end of the load.

Whenever you need to engage IBM DB2 support staff for an autoloader problem, you should provide all the available autoloader log files as above and load message files if there is any. The load temporary files are only needed if you are not able to use RESTART or TERMINATE to bring tablespaces back online after the autoloader failure.

The autoloader's multiple transaction model

The autoloader (**db2atld**) is a multi-thread CLI and SQC application that can retrieve the data file from remote system, partition the data in parallel, and load the data simultaneously on the related

database partitions. It is based on the Multiple Transaction Model, that is, it requires one connection per loading partition. The setup cost is potentially significant, considering a table across a hundred or more partition system.

There is a limitation in 32-bit DB2 applications on AIX, that is, no more than nine multiple logical nodes (MLNs) on the same physical node can be connected from the same DB2 application. Because of this multiple transaction model, the autoloader application is expected to fail with the SQL1224 error when the loaded table is across more than nine logical partitions residing on the same physical node.

You might simulate the SQL1224 error with the following steps:

```
Stop the DB2 instance:
% db2stop

Change the port range in the /etc/services file to allow ten database
partitions to be created on the same physical node:
% vi /etc/services
DB2_db2inst1      26176/tcp
DB2_db2inst1_END     26185/tcp

Change the db2nodes.cfg file by defining ten partitions on the same
physical node:
% vi sqllib/db2nodes.cfg
```

```
 0   af01n002   0   a_sw_002
 1   af01n002   1   a_sw_002
 2   af01n002   2   a_sw_002
 3   af01n002   3   a_sw_002
 4   af01n002   4   a_sw_002
 5   af01n002   5   a_sw_002
 6   af01n002   6   a_sw_002
 7   af01n002   7   a_sw_002
 8   af01n002   8   a_sw_002
 9   af01n002   9   a_sw_002
```

```
Start the DB2 instance:
% db2start

Recreate the sample database across ten partitions:
% db2 drop db sample
% db2sampl /database

Edit an autoloader configuration file:
atld_staff.cfg
```

```
RELEASE=V7.00
db2 load from staff.del of del insert into staff
DATABASE=sample
AUTHENTICATION=NO
SPLIT_NODES=(1,2,3)
MODE=SPLIT_AND_LOAD
```



```
LOGFILE=./staff_log
CHECK_LEVEL=NOCHECK
TRACE=2
```

```
Execute the autoloader operation:
% db2atld -c atld_staff.cfg
…
SQL1224N  A database agent could not be started to service a request,
or was terminated as a result of a database system shutdown or a
force command. SQLSTATE=55032
```

You can confirm the error in the db2diag.log file:

```
2002-11-18-11.23.48.649179   Instance:db2inst1   Node:000
PID:76182(db2atld)   Appid:*LOCAL.db2inst1.021118162348
database_utilities  DIAG_NOTE   Probe:0
Num nodes being autoloaded: 10, 0

2002-11-18-11.23.48.975913   Instance:db2inst1   Node:000
PID:76182(db2atld)   Appid:
oper_system_services  sqlocshr   Probe:200

errno: 0000 0018
```

The function sqlocshr() connects a process to the shared memory set specified by the input parameter. You can convert the error number from 0000 0018 in hex-decimal to 24 in decimal. The `/usr/include/errno.h` file explains error number 24 as: `Too many open files in system`. The above error is the result of the limitation in 32-bit DB2 applications on AIX addressed earlier, which results from the fact that the application process is attempting to attach to more than eleven shared memory segments. However, there is an AIX operating system limitation of eleven shared memory segments per process. What needs to be done is to bypass the use of the IPC shared memory segment that is attached by each local conncection. To do this, you should set up local TCP/IP loopback for the database catalog as follows:

```
% db2 catalog tcpip node loop00 remote af01n002 server xdb2inst1
% db2 catalog db sample as loop_sample at node loop00
% db2 list db directory
Database 2 entry:
Database alias                  = LOOP_SAMPLE
 Database name                   = SAMPLE
 Node name                       = LOOP00
 Database release level          = 9.00
 Comment                         =
 Directory entry type            = Remote
 Catalog node number             = -1
```

You then need to modify the autoloader configuration with a reference to the new TCPIP database catalog alias name.

```
% vi atld_staff.cfg
…
```



```
DATABASE=loop_sample
…
```

Then you can retry the autoloader job:

```
% db2atld -c atld_staff.cfg
```

Concurrent autoloader operations

Problem: An autoloader operation fails with SQL3805N when there are multiple concurrent autoloader sessions running.

You might simulate the SQL3805 error with the following steps:

Edit two autoloader configuration files as follows:

atld_staff.cfg	**atld_org.cfg**
RELEASE=V7.00 db2 load from staff.del of del insert into staff DATABASE=sample AUTHENTICATION=NO SPLIT_NODES=(1,2,3) MODE=SPLIT_AND_LOAD LOGFILE=./staff_log CHECK_LEVEL=NOCHECK TRACE=2	RELEASE=V7.00 db2 load from org.del of del insert into org DATABASE=sample AUTHENTICATION=NO SPLIT_NODES=(1,2,3) MODE=SPLIT_AND_LOAD LOGFILE=./org_log CHECK_LEVEL=NOCHECK TRACE=2

In Session 1, execute the autoloader operation with loading into the STAFF table:

```
% db2atld -c atld_staff.cfg
```
wait till the load initialization requests start

Then in Session 2, execute the autoloader operation with loading into the ORG table:

```
% db2atld -c atld_org.cfg
Utility program: "db2atld". Version: "07020".
Start reading autoloader configuration file: atld_org.cfg.
Finish reading autoloader configuration file: atld_org.cfg.
Start initializing autoloader process.
Finish initializing autoloader process.
The AutoLoader is now issuing all LOAD requests.
The AutoLoader is waiting for all LOAD operations to complete.
SQL3805N  The state of the application or of one or more table spaces
for the table specified prohibits the loadapi action or quiescemode "2".
Reason code ="1".
```

…

During the load initialization, db2atld sends load requests to each output partition, and quiesce the tablespace exclusively on the output partitions. Both STAFF and ORG tables reside in the same tablespace, USERSPACE1. So after the autoloader job with the STAFF table quiesces the tablespace exclusively, the subsequent autoloader job with the ORG table fails with SQL3805N rc=1, where rc=1 means the state of one of the table spaces for the table specified prohibits the loadapi action or quiescemode. You should make sure that multiple concurrent autoloader sessions are allowed only when the loaded tables are in different tablespaces.

To get around this SQL3805N error, you can resubmit the autoloader job with the ORG table after the autoloader into the STAFF table finishes successfully.

```
% db2atld -c atld_org.cfg
```

Bring tablespace online

There are a lot of situations when you want to cancel a running autoloader operation. And then the related tablespace is left in an abnormal state, which prevents users to access the tables in that tablespace.

You might practice how to bring the tablespace back online after an autoloader operation is interrupted with the following steps:

Execute the autoloader operation loading into the STAFF table:

```
% db2atld -c atld_staff.cfg
```

After both load and split initializations finish, you may use CTR+Z to cancel this autoloader job. Then on each output partition, verify the tablespace state with the `list tablespaces command`:

```
% db2 terminate
% export DB2NODE=XXX
```
where *XXX* is the output partition number
```
% db2 connect to sample
% db2 list tablespaces show detail
Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x000c
   Detailed explanation:
     Quiesced: EXCLUSIVE
     Load pending
Total pages                          = 190
 Useable pages                        = 190
 Used pages                           = 190
```



```
Free pages                          = Not applicable
High water mark (pages)             = Not applicable
Page size (bytes)                   = 4096
Extent size (pages)                 = 32
Prefetch size (pages)               = 32
Number of containers                = 1
State change tablespace ID          = 2
State change object ID              = 3
Number of quiescers                 = 1
  Quiescer 1:
    Tablespace ID                   = 2
    Object ID                       = 3
```

Since USERSPACE1 is in Quiesce Exclusive and load pending state, you can tell that there is a load failure during the load phase into a table residing in USERSPACE1. If you don't know which table is loaded, you can query the SYSCAT.TABLES to verify the table name based on the quiescer information as follows:

```
% db2 "select tabname from syscat.tables where tbspaceid=2 and tableid=3"
TABNAME
--------------------------------
STAFF

  1 record(s) selected.
```

You can then use either RESTART to continue the autoloader operation or TERMINATE to terminate the failing operation and bring the tablespace back to normal state. One thing you should keep in mind is that RESTART and TERMINATE can only be used within the `db2atld` command, and cannot be specified within the load command in the configuration file.

```
% db2atld -c atld_staff.cfg -restart
```

or

```
% db2atld -c atld_staff.cfg -terminate
```

It is possible that only some of the partitions require a restart or terminate. The partitions that will be restarted or terminated as expected; the others will return an SQL3805 (incorrect tablespace state), which is normal.

Socket usage by autoloader

The autoloader uses direct TCP/IP communication using sockets for all data transfer required during SPLIT and LOAD processes. Moreover, since db2atld is an application that doesn't have the access to DB2 engine internal structures, all the control messages between all the processes across all the related partitions are passed through sockets as well.

When db2atld uses sockets as internal communications channels (as opposed to named pipes), it uses the following priority sequence to determine the TCP/IP port range:

- The DB2ATLD_PORTS DB2 registry variable which specifies the range as: <lower-port-number>:<higher-port-number>
- The PORTS AutoLoader configuration parameter which specifies the range as: <lower-port-number>:<higher-port-number>
- The default range of 6063 down to 6000

Ports used per autoloader job out of the port range are as follows:

```
The db2atld process on the coordinate partition reserves (2 + 2 * (# of splitters) + (# of output nodes)) ports
The db2split process on each split partition reserves (3 + (# of output nodes)) ports
The loader process On each loading partition reserves (2 + (# of splitters)) ports
```

If not enough ports are reserved for an autoloader operation, the job fails with communication errors. You might simulate such problem with the following steps:

Edit an autoloader configuration file similar to the following one:

new_atld_staff.cfg

```
RELEASE=V7.00
db2 load from staff.del of del insert into staff
DATABASE=sample
AUTHENTICATION=NO
SPLIT_NODES=(1,2,3)
MODE=SPLIT_AND_LOAD
LOGFILE=./newstaff_log
PORTS=6000:6001
CHECK_LEVEL=NOCHECK
TRACE=2
```

Execute the autoloader operation, and see the following output at the console:

```
% db2atld -c new_atld_staff.cfg
...
The AutoLoader is now issuing all split requests.
SQL6555N  The AutoLoader utility encountered an unexpected communications
error.
Start db2split on node "1" in background.
Start db2split on node "3" in background.
Start db2split on node "2" in background.
The remote execution of the splitter utility on partition "2" finished
with remote execution code "-6555".
The remote execution of the splitter utility on partition "3" finished
with remote execution code "-6555".
SQL6555N  The AutoLoader utility encountered an unexpected communications
error.
```



```
SQL6555N  The AutoLoader utility encountered an unexpected communications
error.
The AutoLoader is waiting for all LOAD operations to complete.
The remote execution of the splitter utility on partition "1" finished
with
remote execution code "-6555".
SQL6555N  The AutoLoader utility encountered an unexpected communications
error.
...
```

From the above autoloader console output, you can tell that db2 splitters on all three splitting partitions, 1, 2, and 3, fail with `SQL6555N: The Autoloader utility encountered an unexpected communications error.`

In order to get around that problem, you need to calculate the number of sockets needed by this autoloader operation based on the three formulas introduced earlier, and then set the correct number of ports through the DB2ATLD_PORTS registry variable, the PORTS parameter in the autoloader configuration file, or using the default value.

The following workaround is just to use the default port range of 6063 to 6000:

Use the vi editor command to delete the line of PORTS=6000:6001 from the autoloader configuration file:

```
% vi new_atld_staff.cfg
```

delete the line of PORTS=6000:6001

Terminate the previous failing autoloader operation:

```
% db2atld -c new_atld_staff.cfg -terminate
```

Redo the autoloader operationb:

```
% db2atld -c new_atld_staff.cfg
```

Autoloader coredump

Sometimes the autoloader may fail with a core dump. To analyze the autoloader's core file, you need to run dbx or gdb on the machine where the core file was generated as follows:

```
% dbx /home/db2inst1/sqllib/bin/db2atld core

The map command lists all the Loaded libraries when the core file was
generated:
(dbx) map

db2atld is multi-threaded. The thread command lists all the threads of
the
failing application, and tells you which thread triggers the core dump as
well:
(dbx) thread
```

```
Let's assume that the thread that caused the core dump is #3, and then
you can use the following dbx commands to find out the stack traceback of
the failing thread and the information of its registers:

Switch to the failing thread based on the thread number:
(dbx) thread current 3

The where command shows the stack traceback:
(dbx) where

The registers command shows the information of the failing thread's
registers:
(dbx) registers
```

Once you identify the trapped function from the failing thread's stack traceback, search the DB2 APAR database to verify whether it matches any known DB2 defect. If not, you should engage the IBM DB2 Support Team with the following information:

- the above dbx output
- errpt -a (system log file)
- the autoloader configuration file
- lslpp -ah (which shows db2level and oslevel as well)

Understanding problems with adding or removing nodes and redistributing data

Adding partitions - temporary tablespace consideration

The **ADD NODE** command is used to add more partitions in a partitioned database environment. It has three options associated with temporary tablespace creation on the newly added partition:

- No indication - allows you to default to use the temporary tablespace container definitions from the catalog partition
- Like Node - allows you to create temporary tablespaces with containers like another existing database partitions
- Without Tablespaces - allows you to indicate that you do not want the temporary tablespace containers created. (In this case, later on you must issue an ALTER TABLESPACE statement to add temporary tablespace containers to the database partition before the database can be used.)

During the adding partition operation, you should be careful about the temporary tablespace usage in the database. Otherwise, the operation may fail with `SQL6073N: Add Node operation failed. SQLCODE = "-294"`, where SQL0294 means: `The container is already in use`.

You might simulate the problem with the following steps:

First create some database activity by importing some data into the STAFF table:

```
% db2 connect to sample
% db2 import from staff.del of del insert into staff
```

You can use the following command to verify the data distribution across partitions:

```
% db2 "select nodenumber(ID),count(*) from staff group by nodenumber(ID)"
1              2
-----------  -----------
0                605
1                495
2                275
3                550

  4 record(s) selected.
```

Stop the DB2 instance, and use the vi editor command to modify the db2nodes.cfg file by defining one more partition on the second physical node:

```
% db2stop
% vi sqllib/db2nodes.cfg
```

Partition	Hostname	Port	Netname
0	af01n002	0	a_sw_002
1	af01n002	1	a_sw_002
2	af01n003	0	a_sw_003
3	af01n003	1	a_sw_003
4	**af01n003**	**2**	**a_sw_003**

Start the DB2 instance on the newly added node that is partition 4 in this example:

```
% db2start nodenum 4
```

You will need to run the ADD NODE command for the newly added partition 4, from the physical node where it resides. Before you can do that, however, you should set the DB2NODE registry variable with the new partition number because that physical node has database partitions 2 and 3 defined already:

First, log in the second physical node where Partition 4 resides
Switch to Partition 4:

```
% db2 terminate
% export DB2NODE=4
```

The ADD NODE command creates database partition 4 for all databases currently defined in the MPP server. And at this step you might see the following error message at the console:

```
% db2 add node
SQL6073N  Add Node operation failed.  SQLCODE = "-294".
% db2 ? sql0294
SQL0294N The container is already in use.
```

You can confirm the error message in the db2diag.log file:

```
2002-11-13-18.14.19.136769   Instance:db2inst1   Node:004
PID:78052(db2agent (instance) 4)   Appid:
buffer_pool_services  sqlbSMSAcquireContainer   Probe:818
DIA9999E An internal error occurred. Report the following error code:
"FFFF8139".

2002-11-13-18.14.19.175225   Instance:db2inst1   Node:004
PID:78052(db2agent (instance) 4)   Appid:
buffer_pool_services  sqlbSMSAcquireContainer   Probe:818

Error acquiring container 0 (/database/db2inst1/temp0) for tbsp 3.
Rc = FFFF8139.
```


Now you need to check the temporary tablespace container definitions on the catalog partition 0 and the other two existing database partitions 2 and 3 on the same physical node where the new partition 4 will be added.

Start the DB2 instance on all existing database partitions:

```
% db2start
```

Switch to the catalog partition 0:

```
% db2 terminate
% export DB2NODE=0
```

Get the MYTEMP tablespace's ID:

```
% db2 list tablespaces
…
Tablespace ID                        = 3
Name                                 = MYTEMP
…
```

Check MYTEMP's container definitions on the catalog partition 0:

```
% db2 list tablespace containers for 3 show detail
            Tablespace Containers for Tablespace 3

 Container ID                         = 0
 Name                                 = /database/db2inst1/temp0
 Type                                 = Path
 Total pages                          = 1
 Useable pages                        = 1
 Accessible                           = Yes
```

You can use the similar method to further find out that MYTEMP's container definition on partition 2 is /database/db2inst1/temp0. Since you issued the ADD NODE command with no temporary tablespace indication, DB2 tried to use the temporary tablespace container definition from the catalog partition 0 (/database/db2inst1/temp0) for the new partition 4 on the second physical node while the database partition 2 is using that container path already.

You can work around this problem as follows:

Switch to the newly added partition 4:

```
% db2 terminate
% export DB2NODE=4
```

Add node without creating the temporary tablespace containers:

```
% db2 add node without tablespaces
```

Use the ALTER TABLESPACE statement to add temporary tablespace containers to this newly added database partition:

```
% db2 connect to sample
% db2 "alter tablespace mytemp add ('/database/db2inst1/temp2') on node
(4)"
% db2 "alter tablespace tempspace1 add ('/database/db2inst1/NODE0004
```



```
/SQL00001/SQLT0001.0') on node (4)"
```

After you successfully add a new database partition, nodegroup IBMTEMPGROUP's definition has been changed to include the new database partition. However, any other nodegroups remain unchanged. To use the new database partition in those nodegroups, you must add the new database partition to them. For example:

Update nodegroup definition with the like node method to define container definitions on the new partition similar to those of existing containers definitions on the specified partition:

```
% db2 "alter nodegroup ibmdefaultgroup add nodes (4) like node 3"
SQL1759W  Redistribute nodegroup is required to change data
partitioning for objects in nodegroup "IBMDEFAULTGROUP" to include
some added nodes or exclude some dropped nodes.  SQLSTATE=01618
```

Partition redistribution - log full

After adding a database partition and updating the nodegroup definition, you need to use the **REDISTRIBUTE NODEGROUP** utility to move the data to the appropriate database partitions. Partition redistribution is an insert-delete procedure. DB2 logs each insert and delete SQL on the related partitions. If the log space size is not big enough, it's easy for the partition redistribution operation to fail with an SQL0964 (log full) error when a large number of records need to be redistributed across partitions.

You might simulate the problem with the following steps:

Decrease the log space size in order to easily simulate the problem situation:

```
% db2_all "db2 update db cfg for sample using LOGFILSIZ 4"
% db2_all "db2 update db cfg for sample using LOGPRIMARY 2"
% db2_all "db2 update db cfg for sample using LOGSECOND 0"
```

Recycle the database in order to have the db cfg update take effect:

```
% db2 force applications all
% db2 connect to sample
```

Keep in mind that the REDISTRIBUTE NODEGROUP command needs to be submitted from the catalog node.

Switch to the catalog partition 0:

```
% db2 terminate
%export DB2NODE=0

% db2 connect to sample
% db2 "redistribute nodegroup ibmdefaultgroup uniform"
SQL6064N  SQL error "-964" occurred during data redistribution.
```



```
% db2 ? sql0964
SQL0964C The transaction log for the database is full.
```

You can confirm the error message in the db2diag.log file:

```
2002-11-15-17.49.43.302807   Instance:db2inst1   Node:000
PID:55104(db2agent (SAMPLE) 0)   Appid:*LOCAL.db2inst1.021115223820

data_protection  sqlpgrsp   Probe:50   Database:SAMPLE

Log Full -- active log held by appl. handle 45
End this application by COMMIT, ROLLBACK or FORCE APPLICATION.

2002-11-15-17.49.43.344503   Instance:db2inst1   Node:000
PID:55104(db2agent (SAMPLE) 0)   Appid:*LOCAL.db2inst1.021115223820
data_protection  sqlpWriteLR   Probe:80   Database:SAMPLE
DIA3609C Log file was full.

ZRC=FFFFD509
…

2002-11-15-18.32.01.151874   Instance:db2inst1   Node:003
PID:87008(db2agntp (SAMPLE) 3)   Appid:*LOCAL.db2inst1.021115231508
data_protection  sqlpgrsp   Probe:20   Database:SAMPLE

Warning: active log held by dirty pages.
Decrease softmax and/or increase num_iocleaners.

Data Title:SQLRDRDT 170: rc  PID:56846 Node:000
ffff 876d                                        .m

Data Title:SQLCA PID:56846 Node:000
 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: -964   sqlerrml: 0
 sqlerrmc:
 sqlerrp : sqlridel
 sqlerrd : (1) 0xFFFFD509      (2) 0x00000009      (3) 0x00000000
           (4) 0x00000000      (5) 0x00000000      (6) 0x00000000
 sqlwarn : (1)      (2)      (3)      (4)        (5)       (6)
           (7)      (8)      (9)      (10)       (11)
 sqlstate:
```

From the db2diag.log file, you can tell that partition 0 and 3 hit the log full error. You now need to increase the log space size on those two partitions. You can either increase the log file size or the number of the log files. The following example chooses to increase the log file size on partition 0 and 3:

Switch to partition 0:

```
% db2 terminate
% export DB2NODE=0
```

Increase the log space size by increasing the LOGFILSIZ db cfg parameter:

```
% db2 update db cfg for sample using LOGFILSIZ 1000
```

Switch to partition 3:

```
% db2 terminate
% export DB2NODE=3
```

Increase the log space size by increasing the LOGFILSIZ db cfg parameter:

```
% db2 update db cfg for sample using LOGFILSIZ 1000
```

Then you can retry the redistribute operation:

```
% db2 redistribute nodegroup ibmdefaultgroup uniform
SQL6056N  Nodegroup cannot be redistributed.  Reason code = "2".
% db2 ? sql6056
…
(2) A previous redistribution operation did not complete successfully.
```

If the data redistribution operation fails, some tables may be redistributed while others are not. This occurs because data redistribution is performed one table at a time. So you cannot recover from the failing redistribution operation by simply reissuing that command. You have two options for recovery:

1. Use the CONTINUE option to continue the operation to redistribute the remaining tables.

   ```
   % db2 redistribute nodegroup ibmdefaultgroup continue
   ```

2. Use the ROLLBACK option to undo the redistribution and set the redistributed tables back to their original state.

   ```
   % db2 redistribute nodegroup ibmdefaultgroup rollback
   % db2 redistribute nodegroup ibmdefaultgroup uniform
   ```

Dropping a database partition

Before you are able to drop a database partition, you must first ensure that the database partition being dropped is not being used by any database. A common problem with dropping a database partition is that it may fail with `SQL6035W: Node "<node>" is being used by database "<database>"`.

You might practice how to drop a database partition with the following steps:

Create a deadlock event monitor **EVMON1** on partition 4:

```
% db2 terminate
% export DB2NODE=4
% db2 connect to sample
% db2 "create event monitor evmon1 for deadlocks write to file
```



```
'/home/db2inst1/eventmon' maxfiles 3 maxfilesize 1000"
```
where '/home/db2inst1/eventmon' should pre-exist.

Redistribute data for all databases on database partition to be dropped:

```
% db2 "redistribute nodegroup ibmdefaultgroup uniform drop node (4)"
```

To ensure that the database partition is not in use, use the **DROP NODE VERIFY** command:

```
% db2 drop node verify
SQL6035W  Node "4" is being used by database "SAMPLE".
```

You can use the following command to verify whether there is still any nodegroup object defined on that drop pending partition 4:

```
% db2 list nodegroups show detail
```

Since none of the nodegroups are still defined on this partition any more, you may then query the SYSCAT.EVENTMONITORS view to verify whether there is any event monitor object defined on this partition:

```
% db2 "select evmonname, nodenum from syscat.eventmonitors"
EVMONNAME          NODENUM
------------------              -------
EVMON1                    4

  1 record(s) selected.
```

It is confirmed that the event monitor EVMON1 is defined on partition 4. You need to drop this event monitor before this partition can be dropped.

```
% db2 "drop event monitor evmon1"
% db2 drop node verify
SQL6034W  Node "4" is not being used by any databases.
```

You can now drop the database partition using the db2stop command with the **DROP NODENUM** option:

```
% db2stop drop nodenum 4
SQL6076W  Warning! This command will remove all database files on the
node for this instance.  Before continuing, ensure that there is no
user data on this node by running the DROP NODE VERIFY command.
Do you want to continue ? (y/n)y
```

At last you can view the db2nodes.cfg to confirm that partition 4 is dropped successfully:

```
% more db2nodes.cfg
0 af01n002 0 a_sw_002
1 af01n002 1 a_sw_002
```



```
2 af01n003 0 a_sw_003
3 af01n003 1 a_sw_003
```


Understanding DB2 EEE recovery problems

Crash recovery - missing active log files

Whenever transactions against the database are interrupted unexpectedly, the database becomes inconsistent and unusable. Crash recovery is also called restart recovery, which ensures that database is brought to a consistent and usable state following abnormal termination.

Sometimes when a crash occurs (kill, trap, etc.) and DB2 is brought down abnormally, you may get an SQL1036C error on the first connection to the database.

You might simulate the problem with the following steps:

Open two sessions:
In Session 1
Turn off the session's auto-commit feature:

```
% export DB2OPTIONS=+c
```

Use the following command to verify that the auto-commit is turned off:

```
% db2 list command options
-c    Auto-Commit            OFF
```

Create some activity to the database. For example,

```
% db2 import from staff.ixf of ixf create into new_staff
```

In Session 2
Use the `ps` command to randomly choose a DB2 process associated with your instance.

```
% ps -elf | grep db2inst1
…
240401 A db2inst1 114008      1   0  60 20 2bcd0  1248          17:55:22
- 0:00 /home/db2inst1/sqllib/bin/db2bp 37930 5
…
```

Simulate an abnormal instance shut down by killing a DB2 process. Keep in mind when the instance is up and running, no DB2 process is allowed to be killed; otherwise, the instance is expected to crash.

```
% kill -9 114008
% db2 connect to sample
SQL1032N  No start database manager command was issued.  SQLSTATE=57019
It indicates that db2 instance crashes.
```


Simulate a situation of missing an active log file:

```
% cd /database/db2inst1/NODE0000/SQL00001/SQLOGDIR
% mv S0000000.LOG S0000000.LOG.bak
```

Since the instance shut down abnormally, you need to manually clean up the instance resources before you can restart the instance by issuing the following commands on each physical node:

```
% db2_kill
% ps -elf | grep db2inst1
Please repeat the step "kill -9 <pid of db2 process>" to clean up all
the left over db2 processes associated with your instance.
% ipclean
% ipcs | grep db2inst1
Please repeat the step "ipcrm -q/m/s <ipc resource id>" to clean up all
the left over ipc resources associated with your instance.
```

Now you can start the DB2 instance, and connect to your database. And the first connection to the database, which invokes the crash recovery process, fails with the SQL1036C error:

```
% db2start
% db2 connect to sample
SQL1036C  An I/O error occurred while accessing the database.
SQLSTATE=58030
```

You can confirm the error in the db2diag.log file:

```
2002-11-11-18.07.08.427122   Instance:db2inst1   Node:000
PID:110314(db2loggr 0)   Appid:none
data_protection  sqlpgicl   Probe:130
DIA3711C A file "" could not be found.

ZRC=FFFFE60A

2002-11-11-18.07.08.531036   Instance:db2inst1   Node:000
PID:110314(db2loggr 0)   Appid:none
data_protection  sqlpgilt   Probe:50
DIA3711C A file "" could not be found.

ZRC=FFFFE60A

2002-11-11-18.07.08.614105   Instance:db2inst1   Node:000
PID:110314(db2loggr 0)   Appid:none
data_protection  sqlpgasn   Probe:400

Logging can not continue due to an error. 0000 0000
```


So you can tell that the logging cannot continue due to a missing file. In order to find out which file db2logger is looking for, you can then take a db2 trace as follows:

```
% db2trc on -l 8000000 -e -1
% db2 connect to sample
% db2trc dump trace.dmp
% db2trc off
% db2trc flw trace.dmp trace.flw
% db2trc fmt trace.dmp trace.fmt
```

trace.fmt reveals the following information:

```
2531    DB2 cei_data      oper_system_services sqloopenp (1.25.15.97)
        pid 87118; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 2
        2f64 6174 6162 6173 652f 7869 616f 6d65 /database/db2inst
        6977 2f4e 4f44 4530 3030 302f 5351 4c30 1/NODE0000/SQL0
        3030 3034 2f53 514c 4f47 4449 522f 5330 0004/SQLOGDIR/S0
        3030 3030 3030 2e4c 4f47 0000 005c 0000 000000.LOG...\..
        0180                                    ..
…
2533    DB2 cei_errcode   oper_system_services sqloopenp (1.6.15.97)
        pid 87118; tid 1; node 0; cpid 0; sec 0; nsec 0; tpoint 254
        rc = 0xffffe60a = -6646 = SQLO_FNEX
```

The above two trace entries reveal that /database/db2inst1/NODE0000/SQL00001/SQLOGDIR/S00000000.LOG doesn't not exist (SQLO_FNEX).

You can then further check out the active log directory on partition 0:

```
% ls -al /database/db2inst1/NODE0000/SQL00001/SQLOGDIR
total 112
drwxr-s---   2 db2inst1 sys          512 Nov 11 18:17 ./
drwxr-s---   7 db2inst1 sys          512 Nov 11 18:17 ../
-rw-------   1 db2inst1 sys        24576 Nov 11 17:22 S0000001.LOG
-rw-------   1 db2inst1 sys        24576 Nov 11 17:22 S0000002.LOG
```

Now you have the following options to further pursue this problem situation:

- If you can get back the missing active log file, put the missing log file back in the active log directory.
- If you can't find the missing active log file any more, you may gather the following diagnostic data and engage IBM DB2 Support Team:
    - db2nodes.cfg
    - db2diag.log on the partition where the connection fails
    - the list of log files on the failing database partition
    - the log control head file (SQLOGCTL.LFH) on the failing database partition

- db2 trace of failing database connection command

Version recovery - SQL1035 (database in use) when taking offline backups of all partitions in parallel

The version recovery method requires loading a backup copy of the database. The database will be restored to exactly the same state that it was in when it was backed up. Using the version recovery method, you must schedule and perform full backups of the database on a regular basis.

You may have the following experience:
All database partitions are inactive at the moment. However, offline backup of all partitions in parallel fails with `SQL1035N (The database is currently in use.)` on most of the partitions.

You try to take offline backup of all partitions in parallel as follows:

```
% db2 force applications all
% db2_all ";db2 backup database sample to /database/backup"
af01n002: SQL1035N  The database is currently in use.  SQLSTATE=57019
af01n002: db2 backup database ... completed rc=4

af01n003: SQL1035N  The database is currently in use.  SQLSTATE=57019
af01n003: db2 backup database ... completed rc=4

af01n003: SQL1035N  The database is currently in use.  SQLSTATE=57019
af01n003: db2 backup database ... completed rc=4

af01n002:
af01n002: Backup successful. The timestamp for this backup image is :
20021029190857
af01n002:
af01n002: db2 backup database ... completed ok
```

You can confirm the error in the db2diag.log file:

```
2002-10-29-19.08.58.628733   Instance:db2inst1   Node:001
PID:41494(db2agent (SAMPLE) 1)   Appid:*N1.db2inst1.021030000858
relation_data_serv  sqlrrain   Probe:70   Database:SAMPLE
DIA9999E An internal error occurred. Report the following error code :
"0xFFFF876D".

Data Title:SQLCA PID:41494 Node:001
 sqlcaid : SQLCA     sqlcabc: 136   sqlcode: -1035   sqlerrml: 0
 sqlerrmc:
 sqlerrp : SQLESRSU
 sqlerrd : (1) 0x00000000      (2) 0x00000000      (3) 0x00000000
           (4) 0x00000000      (5) 0x00000000      (6) 0x00000000
 sqlwarn : (1)      (2)      (3)      (4)        (5)       (6)
           (7)      (8)      (9)      (10)       (11)
 sqlstate:
```


This occurs because the database backup utility requires exclusive access to the catalog partition. To properly backup the database in parallel, you may use the following commands:

```
% db2_all "<<+0<; db2 backup database sample to /database/backup"
% db2_all "<<-0<; db2 backup database sample to /database/backup"
```

This assumes that CATALOG PARTITION is partition 0. It is also a good illustration of why the CATALOG PARTITION should contain catalog data ONLY.

Version recovery - log sync mismatch

Another common problem with the version recovery method is log sync mismatch. And you might simulate the problem with the following steps:

Create some activity to the database. For example,

```
% db2 connect to sample
% db2 import from staff.ixf of ixf create into new_staff
```

Simulate a disk failure with a tablespace on the third partition, Partition 2:

```
% db2stop force
% cd /database/db2inst1/NODE0003/SQL00001
% mv SQLT0002.0 SQLT2.0
% db2start
```

Recover the disk failure on Partition 2 by restoring from a backup at this partition only:

```
% db2 terminate
% export DB2NODE=2
% db2 restore database sample from /database/backup
```

After the restore, you might see the following error from the console when you try to connect to the database:

```
% db2 connect to sample
SQL1471N  Cannot connect to database "SAMPLE" on node "2" because the
database on this node is not synchronized with catalog node.
SQLSTATE=0800
```

You can confirm the error in the db2diag.log file:

```
2002-10-29-22.43.45.091036   Instance:db2inst1   Node:000
PID:41532(db2agntp (SAMPLE) 0)   Appid:*N2.db2inst1.021030034343
data_protection  sqlpLogSync   Probe:100   Database:SAMPLE
```

```
Mismatch of record (1, 0) from node 2 with record (3, 0) on catalog node
0

2002-10-29-22.43.44.590874   Instance:db2inst1   Node:002
PID:43668(db2agent (SAMPLE) 2)   Appid:*N2.db2inst1.021030034343
data_protection  sqlpLSrequestor   Probe:90   Database:SAMPLE

Log sync failed with sqlca 5351 4c43 4120 2020 0000 0088 ffff fa41
SQLCA....  ·A
0008 5341 4d50 4c45 ff32 0000 0000 0000       ..SAMPLE 2......
0000 0000 0000 0000 0000 0000 0000 0000       ................
0000 0000 0000 0000 0000 0000 0000 0000       ................
0000 0000 0000 0000 0000 0000 0000 0000       ................
0000 0000 0000 0000 5351 4c50 474c 5335       ........SQLPGLS5
0000 0000 0000 0000 0000 0000 0000 0000       ................
0000 0000 0000 0000 2020 2020 2020 2020       ........
2020 2020 2020 2020

2002-10-29-22.43.44.789822   Instance:db2inst1   Node:002
PID:43668(db2agent (SAMPLE) 2)   Appid:*N2.db2inst1.021030034343
data_protection  sqlpgint   Probe:310   Database:SAMPLE

Catalog node and this node not in sync.  SQLCODE -1471
```

This is called a log sync mismatch. In a partitioned database environment, the database is located across many database partition servers (or nodes). For a database with circular logging in a version recovery scenario, you must restore all partitions, and the backup images that you use for the restore database operation must all have been taken at the same time. (Each database partition is backed up and restored separately.)

In this case, to recover properly, you need to restore the backup images from all partitions as follows:

Perform the restore to the catalog partition first:

```
%db2_all "<<+0<; db2 restore database sample from /database/backup"
```

Issue the restore request to the other partitions in parallel

```
%db2_all "<<-0<; db2 restore database sample from /database/backup"
```

Database roll-forward recovery

With roll-forward recovery, you can roll the database forward (that is, past the time when the backup image was taken) by using the active and archived logs to either a specific point in time, or to the end of the active logs.

You might practice the database roll-forward recovery scenario with the following steps:

You can enable archive logging by turning on the LOGRETAIN db cfg parameter as follows:


```
% db2_all "db2 update db cfg for sample using logretain on"
% db2stop
% db2start
% db2 connect to sample
```

Enabling or disabling log retention logging establishes a new recovery point for the database. So the database is placed in BACKUP PENDING state. DB2 requires an offline backup of your database to establish this new recovery point and get the database out of BACKUP PENDING state.

```
% db2_all "db2 backup db sample to /database/backup"
```

Simulate a disk failure on the second partition, Partition 1:

```
% db2stop force
% cd /database/db2inst1/NODE0001/SQL00001
% mv SQLOGCTL.LFH SQLOGCTL.LFH.bak
% db2start
% db2 terminate
% export DB2NODE=1
% db2 connect to sample
SQL1036C  An I/O error occurred while accessing the database.
SQLSTATE=58030
```

You can confirm the error in the db2diag.log file:

```
2002-11-20-23.50.54.251116   Instance:db2inst1   Node:001
PID:126016(db2agent (SAMPLE) 1)   Appid:*LOCAL.db2inst1.021121044501
data_protection  sqlpsize   Probe:20   Database:SAMPLE
DIA3711C A file "" could not be found.

ZRC=0xFFFFE60A
```

You decide to use restore to recover the disk failure. Since the restore operation needs to access the SQLOGCTL.LFH log control head file as well, the restore command then fails with the following error message shown from the console:

```
% db2 restore db sample from /database/backup
SQL1036C  An I/O error occurred while accessing the database.
SQLSTATE=58030
```

In this case, you might continue the recovery by recreating the damaged database partition. Prior to dropping the database at the damaged node, you may want to get the current database configuration file for that partition, so it can be used as reference to update the database configuration file after you create the database at the node.

```
% db2 get db cfg for sample > db.cfg.1
% db2 drop db sample at node
```

And then you need to recreate that newly dropped node with the command of **CREATE DATABASE … AT NODE**. The database is to be created only on the partition where that command is issued. It is not intended for general use. It should be used with RESTORE DATABASE if the database partition was damaged and must be recreated.

```
% db2 create db sample at node
% db2 update db cfg for sample using LOGRETAIN on
% db2 restore db sample from /database/backup
```

The database partition is in roll-forward pending state after it is successfully restored. In a partitioned database environment, the roll-forward command can only be invoked from the catalog partition.

```
% db2 terminate
% export DB2NODE=0
```

In a partitioned database environment, the database is located across many database partitions. If you are performing point-in-time roll-forward recovery, all database partitions must be rolled forward to ensure that all partitions are at the same level. If you need to restore a single database partition, you can perform roll-forward recovery to the end of the logs to bring it up to the same level as the other partitions in the database.

A database roll-forward operation to the end of logs affects the partitions that are specified. If no partitions are specified, it affects all partitions that are listed in the db2nodes.cfg file; if roll-forward recovery is not needed on a particular partition, that partition is ignored.

In this case you might choose to roll forward to the end of logs:

```
% db2 rollforward db sample to end of logs and complete
SQL1261N  Database "SAMPLE" is not in rollforward pending state on
node(s)
"0,2,3", so it does not need to be rolled forward on these nodes.
```

Tablespace roll-forward recovery

You might practice the tablespace roll-forward recovery scenario with the following steps:

Simulate a disk failure on USERSPACE1 on Partition 1:

```
% db2stop force
% cd /database/db2inst1/NODE0001/SQL00001
% mv SQLT0002.0 SQLT2.0

% db2start
% db2 terminate
% export DB2NODE=1
% db2 connect to sample
```



```
% db2 list tablespaces
Tablespace ID                        = 2
Name                                 = USERSPACE1
Type                                 = System managed space
Contents                             = Any data
State                                = 0x4000
  Detailed explanation:
    Offline
```

You decide to use tablespace level restore to recover the disk failure. You restore USERSPACE1 on Partition 1 only, and then roll forward as follows:

```
% db2 terminate
% export DB2NODE=1
% db2 "restore db sample tablespace (userspace1) from /database/backup"
% db2 terminate
% export DB2NODE=0
% db2 "rollforward db sample to end of  logs tablespace (USERSPACE1)"
SQL4908N  The table space list specified for roll-forward recovery on
database "SAMPLE" is invalid on node(s) "0,2,3".
```

You can confirm the error messages in the db2diag.log file:

```
2002-11-23-20.17.21.849383   Instance:db2inst1   Node:002
PID:342218(db2agntp (SAMPLE) 2)   Appid:*N0.db2inst1.021124011720
recovery_manager  sqlpCheckRfwdState   Probe:40   Database:SAMPLE
DIA9999E An internal error occurred. Report the following error code :
"0xFFFF85C3".
…
2002-11-23-20.17.22.833553   Instance:db2inst1   Node:000
PID:623774(db2agntp (SAMPLE) 0)   Appid:*N0.db2inst1.021124011720
recovery_manager  sqlpCheckRfwdState   Probe:40   Database:SAMPLE
DIA9999E An internal error occurred. Report the following error code :
"0xFFFF85C3".
…
2002-11-23-20.17.23.030632   Instance:db2inst1   Node:003
PID:59072(db2agntp (SAMPLE) 3)   Appid:*N0.db2inst1.021124011720
recovery_manager  sqlpCheckRfwdState   Probe:40   Database:SAMPLE
DIA9999E An internal error occurred. Report the following error code :
"0xFFFF85C3".
```

The roll-forward operation fails with 0xFFFF85C3 on Partition 0, 2 and 3. You can look up the error code 0xFFFF85C3 in *DB2 Troubleshooting Guide*. It means `SQL4906:Tablespace rollforward has invalid tablespace set.`

In a partitioned database environment, if you are rolling forward a tablespace to the end of logs, by default the roll-forward database request is sent to all partitions where the tablespace resides. If you don't want to roll the tablespace forward on all partitions, you must supply the list of database partitions. This means the tablespace must be restored on the database partitions on which rollforward performs.

In a partitioned database environment, if you are rolling forward a tablespace to a point in time, you don't have to supply the list of partitions on which the tablespace resides. DB2 submits the roll-forward request to all partitions. This means the tablespace must be restored on all database partitions on which the tablespace resides.

Let's say that now you decide to restore the backup image and apply all the logs on all partition where the tablespace is defined as follows:

```
% db2 terminate
% export DB2NODE=0
% db2 "restore db sample tablespace (userspace1) from /database/backup"
% db2 terminate
% export DB2NODE=2
% db2 "restore db sample tablespace (userspace1) from /database/backup"
% db2 terminate
% export DB2NODE=3
% db2 "restore db sample tablespace (userspace1) from /database/backup"
```

After the tablespace-level restore is performed on all the partitions where USERSPACE1 resides, you may roll forward USERSPACE1 to the end of logs:

```
% db2 terminate
% export DB2NODE=0
% db2 "rollforward db sample to end of logs tablespace (userspace1)"
```

You can then use the following command to verify whether USERSPACE1 recovers from the disk failure:

```
% db2 terminate
% export DB2NODE=1
% db2 connect to sample
% db2 list tablespaces
Tablespace ID                        = 2
Name                                 = USERSPACE1
Type                                 = System managed space
Contents                             = Any data
State                                = 0x0000
  Detailed explanation:
    Normal
```


Summary and resources

Summary

After completing this tutorial you should now have a fundamental understanding of the following troubleshooting skills in a multi-node DB2 environment:

Narrow down which nodes in a multi-node environment are involved in a problem

- What is a good problem description
- What diagnostic data you need to collect
- How to narrow down problematic nodes in a DB2 EEE environment

Understand problems in inter-node communication

- Start and stop DB2 instance in multi-node environment
- Enable Fast Communication Manager (FCM)
- How to handle situations with not enough FCM_NUM_BUFFERS or FCM_NUM_RQB

Diagnose problems with loading data into multiple nodes

- The import utility slow performance and log full situations
- What and where the autoloader diagnostic files are
- The autoloader application's multiple transaction model, concurrency, and socket usage
- How to bring tablespace back online after an autoloader failure
- How to pursue the autoloader core dump situation

Understand problems with adding or removing nodes and redistributing data

- Adding partitions with temporary tablespace consideration
- Partition redistribution resulting in log full situation
- Dropping a database partition only after no database object is defined on the current node

Understand DB2 EEE recovery problems

- Crash recovery with missing active log files
- Version recovery: how to correctly take offline backups on all partitions in parallel, and how to avoid log sync mismatch
- Database and tablespace roll-forward recovery

Resources

Learn more about the commands and utilities described in this tutorial from the following resources, available at the DB2 Technical Support website (http://www.ibm.com/software/data/db2/udb/winos2unix/support):

- *Quick Beginnings: DB2 Enterprise - Extended Edition for UNIX*
- *Administration Guide*

- *Data Recovery and High Availability Guide and Reference*
- *Command Reference*
- *Troubleshooting Guide*
- *DB2 APARs search database*