IBM DB2 Universal Database

# Command Reference

*Version 6*

SC09-2844-00

IBM DB2 Universal Database

# Command Reference

*Version 6*

SC09-2844-00

Before using this information and the product it supports, be sure to read the general information under “Appendix D. Notices” on page 543.

# Contents

# About This Book

This book provides information about the use of system commands and the IBM DB2 Universal Database command line processor (CLP) to execute database administrative functions.

## Who Should Use this Book

It is assumed that the reader has an understanding of database administration and a knowledge of Structured Query Language (SQL).

## How this Book is Structured

This book provides the reference information needed to use the CLP.

The following topics are covered:

**Chapter 1**
Describes the commands that can be entered at an operating system command prompt or in a shell script to access the database manager.

**Chapter 2**
Explains how to invoke and use the command line processor, and describes the CLP options.

**Chapter 3**
Provides a description of all database manager commands.

**Chapter 4**
Provides information on how to use SQL statements from the command line.

**Appendix A.**
Explains the conventions used in syntax diagrams.

**Appendix B**
Explains the conventions used to name objects such as databases and tables.

# Chapter 1. System Commands

This chapter provides information about the commands that can be entered at an operating system command prompt, or in a shell script, to access and maintain the database manager.

**Note:** Slashes (/) in directory paths are specific to UNIX based systems, and are equivalent to back slashes (\) in directory paths on OS/2 and Windows operating systems.

## How the Command Descriptions are Organized

A short description of each command precedes some or all of the following subsections.

### Scope

The command's scope of operation within the instance. In a single-node system, the scope is that single node only. In a multi-node system, it is the collection of all logical nodes defined in the node configuration file, `db2nodes.cfg`.

### Authorization

The authority required to successfully invoke the command.

### Required Connection

One of the following: database, instance, none, or establishes a connection. Indicates whether the function requires a database connection, an instance attachment, or no connection to operate successfully. An explicit connection to the database or attachment to the instance may be required before a particular command can be issued. Commands that require a database connection or an instance attachment can be executed either locally or remotely. Those that require neither cannot be executed remotely; when issued at the client, they affect the client environment only. For information about database connections and instance attachments, see the *Administration Guide*.

### Command Syntax

For information about syntax diagrams, see "Appendix A. How to Read the Syntax Diagrams" on page 521.

### Command Parameters

A description of the parameters available to the command.

### Usage Notes

Other information.

### See Also

A cross-reference to related information.

## db2admin - DB2 Administration Server

This utility is used to manage the DB2 Administration Server. For more information about the DB2 Administration Server, see the *Administration Guide*.

### Authorization

Local administrator on Windows operating systems, or a system administrator on OS/2.

### Required Connection

None

### Command Syntax

```
►►──db2admin──────────────────────────────────────────────────────────────────►

►─┬──────────────────────────────────────────────────────────────┬──────────►◄
  ├─START────────────────────────────────────────────────────────┤
  ├─STOP─────────────────────────────────────────────────────────┤
  ├─CREATE─┬─────────────────────────┬──┬───────────────────────────┬─┤
  │        └─/USER:──user-account──┘  └─/PASSWORD:──user-password─┘ │
  ├─DROP─────────────────────────────────────────────────────────┤
  ├─SETID──user-account──user-password───────────────────────────┤
  └─-?───────────────────────────────────────────────────────────┘
```

### Command Parameters

**Note:** If no parameters are specified, and the DB2 Administration Server exists, this command returns the DB2 Administration Server instance.

**START**
: Start the DB2 Administration Server.

**STOP** Stop the DB2 Administration Server.

**CREATE /USER: user-account /PASSWORD: user-password**
Create the DB2 Administration Server. If a user name and password are specified, the DB2 Administration Server instance will be associated with this user account. If the specified values are not valid, the utility returns an authentication error. The specified user account must be a valid SQL identifier, and must exist in the security database. It is recommended that a user account be specified to ensure that all DB2 Administration Server functions can be accessed.

**DROP** Deletes the DB2 Administration Server instance.

**SETID user-account/user-password**
: Establishes or modifies the user account associated with the DB2 Administration Server instance.

**-?**
: Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## db2adutl - Work with ADSM Archived Images

Permits a user to query, extract, and delete backups, logs, and load copy images saved using ADSM.

For a complete description of this command, see the *Administration Guide*.

## db2advis - DB2 Index Advisor

Advises users on what indexes to create for one or more SQL statements. A group of related SQL statements is known as a *workload*. Users can rank the importance of each statement in a workload, and specify the frequency at which each statement in the workload is to be executed. The recommended indexes for each table, the statistics derived for them, as well as the DDL by which each can be created, are written to a user-created table, ADVISE_INDEX.

### Authorization

Read access to the database. Read and write access to the explain tables.

### Required Connection

None. This command establishes a database connection.

### Command Syntax

```
►►──db2advis──-d──database-name──┬──────────────────────┬─────────────────────────►
                                 ├─-w──workload-name──┤
                                 ├─-s──"statement"────┤
                                 └─-f──filename───────┘

►──┬────────────────────────────┬──┬──────────────────┬──┬────────────────────────┬──►
   └─-a──userid──┬────────────┬─┘  └─-l──disk-limit───┘  └─-t──max-advise-time────┘
                 └─/passwd────┘

►──┬──────┬───────────────────────────────────────────────────────────────────►◄
   └─-h───┘
```

### Command Parameters

**-d database-name**
: Specifies the name of the database to which a connection is to be established.

**-w workload-name**
: Specifies the name of the workload for which indexes are to be advised. This name is used in the ADVISE_WORKLOAD table.

**-s ″statement″**
: Specifies the text of a single SQL statement whose indexes are to be advised. The statement must be enclosed by double quotation marks.

**-f filename**
: Specifies the name of an input file containing one or more SQL

statements. The default is standard input. Identify comment text with two hyphens at the start of each line; that is, `--` *<comment>*. Statements must be delimited by semicolons.

The frequency at which each statement in the workload is to be executed can by changed by inserting the following line into the input file:

```
-- #SET FREQUENCY <x>
```

The frequency can be updated any number of times in the file.

**-a userid/passwd**
Name and password used to connect to the database. The slash (/) must be included if a password is specified.

**-l disk-limit**
Specifies the maximum number of megabytes available for all indexes in the existing schema. The default value is the database manager limit on maximum size of an index per partition (64 GB).

**-t max-advise-time**
Specifies the maximum allowable time, in minutes, to complete the operation. The default value is 10. Unlimited time is specified by a value of zero.

**-h** Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## Examples

In the following example, the utility connects to the PROTOTYPE database, and recommends indexes for the ADDRESSES table without any constraints on the solution:

```
db2advis -d prototype -s "select * from addresses a
   where a.zip in ('93213', '98567', '93412')
   and (company like 'IBM%' or company like '%otus')"
```

In the following example, the utility connects to the PROTOTYPE database, and recommends indexes that will not exceed 53MB for queries in the ADVISE_WORKLOAD table whose workload name is equal to ″production″. The maximum allowable time for finding a solution is 20 minutes.

```
db2advis -d prototype -w production -l 53 -t 20
```

In the following example, the utility connects to the TEST database, and recommends at most five indexes of up to three columns each for each table referenced by any of the queries in the file `myqueries.sql`. No limits are placed on run time, or on the amount of disk space consumed.

```
db2advis -d test -f myqueries.sql -c 3 -n 5
```

In the final example, an input file called `db2advis.in` contains SQL statements and a specification of the frequency at which each statement is to be executed:
`--#SET FREQUENCY 100 SELECT COUNT(*) FROM EMPLOYEE; SELECT * FROM EMPLOYEE WHERE LASTNAME='HAAS'; --#SET FREQUENCY 1 SELECT AVG(BONUS), AVG(SALARY) GROUP BY WORKDEPT ORDER BY WORKDEPT;`

The utility connects to the SAMPLE database, and recommends indexes for each table referenced by the queries in the input file. The maximum allowable time for finding a solution is 5 minutes:

```
db2advis -d sample -f db2advis.in -t 5
```

## Usage Notes

For dynamic SQL statements, the frequency with which statements are executed can be obtained from the monitor as follows:

1. Issue `db2 reset monitor data`. Wait for an appropriate interval of time.
2. Issue `db2 get snapshot for dynamic sql on <database-alias>`.
3. Issue `db2 "insert into advise_workload (select 'myworkload', 0,stmt_text, cast(generate_unique() as char(254)), num_executions, 1, num_executions, 0, 0 from SYSFUN.SQLCACHE_SNAPSHOT)"`.

The default frequency for each SQL statement in a workload is 1, and the default importance is also 1. The generate_unique() function assigns a unique identifier to the statement, which can be updated by the user to be a more meaningful description of that SQL statement.

## db2audit - Audit Facility Administrator Tool

DB2 provides an audit facility to assist in the detection of unknown or unanticipated access to data. The DB2 audit facility generates and permits the maintenance of an audit trail for a series of predefined database events. The records generated from this facility are kept in an audit log file. The analysis of these records can reveal usage patterns which would identify system misuse. Once identified, actions can be taken to reduce or eliminate such system misuse. The audit facility acts at an instance level, recording all instance level activities and database level activities.

Authorized users of the audit facility can control the following actions within the audit facility, using **db2audit**:

- Start recording auditable events within the DB2 instance.
- Stop recording auditable events within the DB2 instance.
- Configure the behavior of the audit facility.
- Select the categories of the auditable events to be recorded.
- Request a description of the current audit configuration.
- Flush any pending audit records from the instance and write them to the audit log.
- Extract audit records by formatting and copying them from the audit log to a flat file or ASCII delimited files. Extraction is done for one of two reasons: In preparation for analysis of log records, or in preparation for pruning of log records.
- Prune audit records from the current audit log.

For a complete description of this command, see the *Administration Guide*.

## db2atld - Autoloader

Autoloader is a tool for partitioning and loading data in an MPP environment. This utility can:

- Transfer data from one system (MVS, for example) to an AIX system (RS/6000 or SP2)
- Partition data in parallel
- Load data simultaneously on corresponding nodes.

For a complete description of this command, see the *Data Movement Utilities Guide and Reference*.

### See Also

"LOAD" on page 338.

## db2batch - Benchmark Tool

Reads SQL statements from either a flat file or standard input, dynamically prepares and describes the statements, and returns an answer set.

### Authorization

One of the following:
- *sysadm*

### Required Connection

None. This command establishes a database connection.

### Command Syntax

```
►►──db2batch──-d──dbname──┬─────────────────┬──┬──────────────────────┬──────────►
                          └─-f──file_name───┘  └─-a──userid/passwd────┘

►──┬────────────────────────────────┬──┬──────────────┬──┬───────────────────────┬──────►
   └─-r──outfile──┬───────────────┬─┘  └─-c──┬─on──┬──┘  └─-i──┬─short────┬──────┘
                  └─,outfile2─────┘          └─off─┘            ├─long─────┤
                                                                └─complete─┘

►──┬─────────────────┬──┬──────────────┬──┬──────────────┬──┬──────────────┬──────►
   └─-o──options─────┘  └─-v──┬─off─┬──┘  └─-s──┬─on──┬──┘  └─-q──┬─off─┬──┘
                              └─on──┘           └─off─┘            └─on──┘

►──┬──────────────────────┬──┬─────────────────────────┬──┬──────┬──────────────►◄
   └─-p──┬─-s──────────┬──┘  └─-cli──┬──────────────┬──┘  └─-h───┘
         └─-t──table───┘             └─cache-size───┘
```

### Command Parameters

**-d dbname**
: An alias name for the database against which SQL statements are to be applied. The default is the value of the **DB2DBDFT** environment variable.

**-f file_name**
: Name of an input file containing SQL statements. The default is standard input.

  Identify comment text with two hyphens at the start of each line, that is, `--` *<comment>*. If it is to be included in the output, mark the comment as follows: `--#COMMENT` *<comment>*.

A *block* is a number of SQL statements that are treated as one, that is, information is collected for all of those statements at once, instead of one at a time. Identify the beginning of a block of queries as follows: `--#BGBLK`. Identify the end of a block of queries as follows: `--#EOBLK`.

Specify one or more control options as follows: `--#SET <control option> <value>`. Valid control options are:

**ROWS_FETCH**
Number of rows to be fetched from the answer set. Valid values are -1 to *n*. The default value is -1 (all rows are to be fetched).

**ROWS_OUT**
Number of fetched rows to be sent to output. Valid values are -1 to *n*. The default value is -1 (all fetched rows are to be sent to output).

**PERF_DETAIL**
Specifies the level of performance information to be returned. Valid values are:

**0** No timing is to be done.

**1** Return elapsed time only.

**2** Return elapsed time and CPU time.

**3** Return a summary of monitoring information.

**4** Return a snapshot for the database manager, the database, the application, and the statement (the latter is returned only if autocommit is off, and single statements, not blocks of statements, are being processed).

**5** Return a snapshot for the database manager, the database, the application, and the statement (the latter is returned only if autocommit is off, and single statements, not blocks of statements, are being processed). Also return a snapshot for the bufferpools, table spaces and FCM (an FCM snapshot is only available in a multi-node environment).

The default value is 1. A value >1 is only valid on DB2 Version 2 servers.

**DELIMITER**
A one- or two-character end-of-statement delimiter. The default value is a semicolon (;).

**SLEEP**
Number of seconds to sleep. Valid values are 1 to *n*.

**PAUSE**
Prompts the user to continue.

**TIMESTAMP**
Generates a time stamp.

**-a userid/passwd**
Name and password used to connect to the database. The slash (/) must be included.

**-r outfile**
An output file that will contain the query results. An optional *outfile2* will contain a results summary. The default is standard output.

**-c** Automatically commit changes resulting from each SQL statement.

**-i** An elapsed time interval (in seconds).

**short** The time taken to open the cursor, complete the fetch, and close the cursor.

**long** The elapsed time from the start of one query to the start of the next query, including pause and sleep times, and command overhead.

**complete**
The time to prepare, execute, and fetch, expressed separately.

**-o options**
Control options. Valid options are:

**f rows_fetch**
Number of rows to be fetched from the answer set. Valid values are -1 to *n*. The default value is -1 (all rows are to be fetched).

**r rows_out**
Number of fetched rows to be sent to output. Valid values are -1 to *n*. The default value is -1 (all fetched rows are to be sent to output).

**p perf_detail**
Specifies the level of performance information to be returned. Valid values are:

| | |
|---|---|
| **0** | No timing is to be done. |
| **1** | Return elapsed time only. |
| **2** | Return elapsed time and CPU time. |

**3** Return a summary of monitoring information.

**4** Return a snapshot for the database manager, the database, the application, and the statement (the latter is returned only if autocommit is off, and single statements, not blocks of statements, are being processed).

**5** Return a snapshot for the database manager, the database, the application, and the statement (the latter is returned only if autocommit is off, and single statements, not blocks of statements, are being processed). Also return a snapshot for the bufferpools, table spaces and FCM (an FCM snapshot is only available in a multi-node environment).

**o query_optimization_class**
Sets the query optimization class. For a description of valid values, see the *Administration Guide*.

**e explain_mode**
Sets the explain mode under which **db2batch** runs. The explain tables must be created prior to using this option. Valid values are:

**0** Run query only (default).

**1** Populate explain tables only. This option populates the explain tables and causes explain snapshots to be taken.

**2** Populate explain tables and run query. This option populates the explain tables and causes explain snapshots to be taken.

**-v** Verbose. Send information to standard error during query processing. The default value is `off`.

**-s** Summary Table. Provide a summary table for each query or block of queries, containing elapsed time (if selected), CPU times (if selected), the rows fetched, and the rows printed. The arithmetic and geometric means for elapsed time and CPU times are provided if they were collected.

**-q** Query output. Valid values are:

**on** Print only the *non-delimited* output of the query.

**off** Print the output of the query and all associated information. This is the default.

**del** Print only the *delimited* output of the query.

**-p** Parallel (MPP only). Valid values are:

**-s** Single table or collocated join query. If this option is specified, the NODENUMBER function will be added to the WHERE clause of the query, and a temporary table will not be created. This option is valid only if the query contains a single table in the FROM clause, or if the tables contained in the FROM clause are collocated.

**-t table** Specifies the table to use for an INSERT INTO statement. If the query contains multiple tables in the FROM clause, and the tables are not collocated, the result set must first be inserted into a temporary table, and then a SELECT from this temporary table must be performed on all nodes.

If neither the -s nor the -t option is specified, the tool creates a temporary table by default.

If a *local* output file is specified (using the **-r** option), the output from each node will go into a separate file with the same name on each node). If a file that is on an NFS-mounted file system is specified, all of the output will go into this file.

**-cli** Run **db2batch** in CLI mode. The default is to use embedded dynamic SQL. The statement memory can be set manually, using the *cache-size* parameter.

**cache-size**
Size of the statement memory, expressed as number of statements. The default value is 25. If the utility encounters an SQL statement that has already been prepared, it will reuse the old plans. This parameter can only be set when **db2batch** is run in CLI mode.

**-h** Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## Examples

For a detailed discussion on the use of **db2batch**, see the *Administration Guide*.

## Usage Notes

Although SQL statements can be up to 32 698 characters in length, no text line in the input file can exceed 3 898 characters, and long statements must be divided among several lines. Statements must be terminated by a delimiter (the default is a semicolon).

SQL statements are executed with the repeatable read (RR) isolation level.

## See Also

"db2sql92 - SQL92 Compliant SQL Statement Processor" on page 77.

## db2bfd - Bind File Description Tool

Displays the contents of a bind file. This utility, which can be used to examine and to verify the SQL statements within a bind file, as well as to display the precompile options used to create the bind file, may be helpful in problem determination related to an application’s bind file.

### Authorization

None

### Required Connection

None

### Command Syntax

```
                           ┌─,─────────┐
                           ▼           │
►►──db2bfd──filespec────────┬─h─┬──────┴──────────────────────────────────────►◄
                            ├─b─┤
                            ├─s─┤
                            └─v─┘
```

### Command Parameters

**filespec**
: Name of the bind file whose contents are to be displayed.

**h**
: Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

**b**
: Display the bind file header.

**s**
: Display the SQL statements.

**v**
: Display the host variable declarations.

## db2cc - Start Control Center

Starts the Control Center. The Control Center is an easy-to-use graphical interface that displays database objects (such as databases, tables, and packages) and their relationship to one another.

### Authorization

*sysadm*

### Command Syntax

```
►►──db2cc──┬─────────────┬──────────────────────────────────────────────►◄
           └─port-number─┘
```

### Command Parameters

**port-number**
: Specifies the reserved DB2JD port number. For example, 6790. The default value is 6789.

### Usage Notes

**Note:** The following commands:

- "GET ADMIN CONFIGURATION" on page 217
- "RESET ADMIN CONFIGURATION" on page 430
- "UPDATE ADMIN CONFIGURATION" on page 494

cannot normally be issued from within the Command Center. This can only be done if the Command Center was started from a DB2 Administration Server instance. To execute these commands from any other instance, use the command line processor (CLP).

For general information about the Control Center, see the *Administration Guide*. Detailed information is provided through the online help facility within the Control Center.

## db2cfexp - Connectivity Configuration Export Tool

Exports connectivity configuration information to an export profile, which can later be imported at another DB2 Universal Database (UDB) workstation instance of similar instance type.

This utility exports connectivity configuration information into a file known as a configuration profile. It is a non-interactive utility that packages all of the configuration information needed to satisfy the requirements of the export options specified. Items that can be exported are:

- Database information (including DCS and ODBC information)
- Node information
- Protocol information
- database manager configuration settings
- UDB registry settings
- Common ODBC/CLI settings.

This utility is especially useful for exporting connectivity configuration information at workstations that do not have the DB2CCA GUI installed, and in situations where multiple similar remote UDB clients are to be installed, configured, and maintained.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Command Syntax

```
                               ┌─TEMPLATE─┐
►►──db2cfexp──filename────┼─BACKUP───┼──────────────────────────────────────►◄
                               └─MAINTAIN─┘
```

### Command Parameters

**filename**
: Specifies the fully qualified name of the target export file. This file is known as a configuration profile.

**TEMPLATE**
: Creates a configuration profile that is used as a template for other instances of the same instance type. The profile includes information about:

- All databases, including related ODBC and DCS information

- All nodes associated with the exported databases
- Common ODBC/CLI settings
- Common client settings in the database manager configuration
- Common client settings in the UDB registry.

**BACKUP**
Creates a configuration profile of the UDB instance for local backup purposes. This profile contains all of the instance configuration information, including information of a specific nature relevant only to this local instance. The profile includes information about:

- All databases including related ODBC and DCS information
- All nodes associated with the exported databases
- Common ODBC/CLI settings
- All settings in the database manager configuration
- All settings in the UDB registry
- All protocol information.

**MAINTAIN**
Creates a configuration profile containing only database- and node-related information for maintaining or updating other instances.

## db2cfimp - Connectivity Configuration Import Tool

Imports connectivity configuration information from a file known as a configuration profile. It is a non-interactive utility that will attempt to import all the information found in the configuration profile.

A configuration profile may contain connectivity items such as:

- Database information (including DCS and ODBC information)
- Node information
- Protocol information
- database manager configuration settings
- Universal Database (UDB) registry settings
- Common ODBC/CLI settings.

This utility can be used to duplicate the connectivity information from another similar instance that was configured previously. It is especially useful on workstations that do not have the DB2CCA GUI tool installed, and in situations where multiple similar remote UDB clients are to be installed, configured, and maintained.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Command Syntax

```
►►──db2cfimp──filename──────────────────────────────────────────────────►◄
```

### Command Parameters

**filename**
: Specifies the fully qualified name of the configuration profile to be imported. Valid import configuration profiles are: profiles created by any UDB connectivity configuration export method, server access profiles, or DDCS Version 2.4 configuration profiles.

## db2cidmg - Remote Database Migration

Supports remote unattended migration in the Configuration, Installation, and Distribution (CID) architecture environment.

### Authorization

One of the following:
- *sysadm*
- *dbadm*

### Command Syntax

```
►►──db2cidmg──┬─database────┬──┬───────────────┬──┬──────┬───────────────────────►◄
              ├─/r=respfile─┤  └─/l1=logfile───┘  └─/b───┘
              └─/e──────────┘
```

### Command Parameters

**database**
: Specifies an alias name for the database which is to be migrated to DB2 Version 5. If not specified, a response file or `/e` must be provided for program invocation. Note that the database alias must be cataloged on the target workstation. However, it can be a local or a remote database.

**/r**
: Specifies a response file to be used for CID migration. The response file is an ASCII file containing a list of databases which are to be migrated. If not specified, a database alias or `/e` must be provided for program invocation.

**/e**
: Indicates that every single database cataloged in the system database directory is to be migrated. If `/e` is not specified, a database alias or a response file must be provided.

**/l1**
: Specifies the path name of the file to which error log information from remote workstations can be copied after the migration process is completed. If more than one database is specified in the response file, the log information for each database migration is appended to the end of the file. Regardless of whether `/l1` is specified or not, a log file with the name `DB2CIDMG.LOG` is generated and kept in the workstation's file system where the database migration has been performed.

**/b**
: Indicates that all packages in the database are to be rebound once migration is complete.

## db2ckmig - Database Pre-migration Tool

Verifies that a database can be migrated. For detailed information about using this tool, see one of the *Quick Beginnings* books.

### Authorization

*sysadm*

### Required Connection

None

### Command Syntax

```
►►──db2ckmig──┬─database─┬──-l filename──┬──────────────┬──┬────────────────┬──────────────►◄
              └─-e───────┘               └─-u userid────┘  └─-p password────┘
```

### Command Parameters

**database**
Specifies an alias name of a database to be scanned.

**-e** Specifies that all local cataloged databases are to be scanned.

**-l** Specifies a log file to keep a list of errors and warnings generated for the scanned database.

**-u** Specifies the user ID of the system administrator.

**-p** Specifies the password of the system administrator's user ID.

### Usage Notes

To verify the state of a database:

1. Logon as the instance owner.
2. Issue the **db2ckmig** command.
3. Check the log file. If it shows errors, see one of the *Quick Beginnings* books for suggested corrective actions.

   **Note:** The log file displays the errors that occur when the **db2ckmig** command is run. Check that the log is empty before continuing with the migration process.

## db2cli - DB2 Interactive CLI

Launches the interactive Call Level Interface environment for design and prototyping in CLI. Located in the `sqllib/samples/cli/` subdirectory of the home directory of the database instance owner.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──db2cli─────────────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Usage Notes

DB2 Interactive CLI consists of a set of commands that can be used to design, prototype, and test CLI function calls. It is a programmers' testing tool provided for the convenience of those who want to use it, and IBM makes no guarantees about its performance. DB2 Interactive CLI is not intended for end users, and so does not have extensive error-checking capabilities.

Two types of commands are supported:

**CLI commands**
Commands that correspond to (and have the same name as) each of the function calls that is supported by IBM CLI

**Support commands**
Commands that do not have an equivalent CLI function.

Commands can be issued interactively, or from within a file. Similarly, command output can be displayed on the terminal, or written to a file. A useful feature of the CLI command driver is the ability to capture all commands that are entered during a session, and to write them to a file, thus creating a *command script* that can be rerun at a later time.

For more information about this utility, see the file `intcli.doc`, which is also located in the `sqllib/samples/cli/` subdirectory of the home directory of the database instance owner.

## db2cmd - Open DB2 Command Window

Opens the CLP-enabled DB2 window, and initializes the DB2 command line environment. Issuing this command is equivalent to clicking on the *DB2 Command Window* icon.

This command is available on Windows NT, Windows 95, and Windows 98.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──db2cmd──┬────┬──────────────────────────────────────────────────►◄
            ├─-c─┤
            ├─-w─┤
            ├─-i─┤
            └─-t─┘
```

### Command Parameters

**-c** Execute the command, and then terminate. For example, ″db2cmd /c dir″ causes the ″dir″ command to be invoked in a command window, and then the command window closes.

**-w** Wait until the cmd.exe process ends. For example, ″db2cmd /c /w dir″ invokes the ″dir″ command, and db2cmd.exe does not end until the command window closes.

**-i** Run the command window, sharing the same console and inheriting file handles. For example, ″db2cmd /c /w /i db2 get dbm cfg > myoutput″ invokes cmd.exe to run the db2 command and to wait for its completion. A new console is not assigned, and stdout is piped to file ″myoutput″.

**-t** Instead of using ″DB2 CLP″ as the title of the command window, inherit the title from the invoking window. This is useful if one wants, for example, to set up an icon with a different title that invokes ″db2cmd /t″.

**Note:** All switches must appear before any commands to be executed. For example: db2cmd /t db2.

## Usage Notes

If DB21061E (″Command line environment not initialized.″) is returned when bringing up the CLP-enabled DB2 window, or running CLP commands on Windows 95 or Windows 98, the operating system may be running out of environment space. Check the `config.sys` file for the SHELL environment setup parameter, and increase its value accordingly. For example:

```
SHELL=C:\COMMAND.COM C:\ /P /E:32768
```

## db2dclgn - Declaration Generator

Generates declarations for a specified database table, eliminating the need to look up those declarations in the documentation. The generated declarations can be modified as necessary. The supported host languages are C/C++, COBOL, JAVA, and FORTRAN.

### Authorization

None

### Required Connection

None

### Command Syntax

```
                                                   ┌────────────┐
                                                   ▼            │
►►──db2dclgn──-d──database-name──-t──table-name────┴─┬────────┬─┴──────────────►◄
                                                      └─option─┘
```

### Command Parameters

**-d database-name**
Specifies the name of the database to which a connection is to be established.

**-t table-name**
Specifies the name of the table from which column information is to be retrieved to generate declarations.

**option** One or more of the following:

**-a action**
Specifies whether declarations are to be added or replaced. Valid values are `ADD` and `REPLACE`. The default value is `ADD`.

**-b lob-var-type**
Specifies the type of variable to be generated for a LOB column. Valid values are:

**LOB (default)**
For example, in C, SQL TYPE is CLOB(5K) x.

**LOCATOR**
For example, in C, SQL TYPE is CLOB_LOCATOR x.

**FILE** For example, in C, SQL TYPE is CLOB_FILE x.

**-c** Specifies whether the column name is to be used as a suffix in the field name when a prefix (**-n**) is specified. If no prefix is specified, this option is ignored. The default behavior is to not use the column name as a suffix, but instead to use the column number, which starts at 1.

**-i**
Specifies whether indicator variables are to be generated. Since host structures are supported in C and COBOL, an indicator table of size equal to the number of columns is generated, whereas for JAVA and FORTRAN, individual indicator variables are generated for each column. The names of the indicator table and the variable are the same as the table name and the column name, respectively, prefixed by ″IND-″ (for COBOL) or ″ind_″ (for the other languages). The default behavior is to not generate indicator variables.

**-l language**
Specifies the host language in which the declarations are to be generated. Valid values are `C`, `COBOL`, `JAVA`, and `FORTRAN`. The default behavior is to generate C declarations, which are also valid for C++.

**-n name**
Specifies a prefix for each of the field names. A prefix must be specified if the **-c** option is used. If it is not specified, the column name is used as the field name.

**-o output-file**
Specifies the name of the output file for the declarations. The default behavior is to use the table name as the base file name, with an extension that reflects the generated host language:

```
.h for C
.cbl for COBOL
.java for JAVA
.f for FORTRAN (UNIX)
.for for FORTRAN (INTEL)
```

**-p password**
Specifies the password to be used to connect to the database. It must be specified if a user ID is specified. The default behavior is to provide no password when establishing a connection.

**-r remarks**
Specifies whether column remarks, if available, are to be used as comments in the declarations, to provide more detailed descriptions of the fields.

**-s structure-name**
Specifies the structure name that is to be generated to group all the fields in the declarations. The default behavior is to use the unqualified table name.

**-u userid**
Specifies the user ID to be used to connect to the database. It must be specified if a password is specified. The default behavior is to provide no user ID when establishing a connection.

**-v**
Specifies whether the status (for example, the connection status) of the utility is to be displayed. The default behavior is to display only error messages.

**-w DBCS-var-type**
Specifies whether sqldbchar or wchar_t is to be used for a GRAPHIC / VARGRAPHIC / DBCLOB column in C.

**-y DBCS-symbol**
Specifies whether G or N is to be used as the DBCS symbol in COBOL.

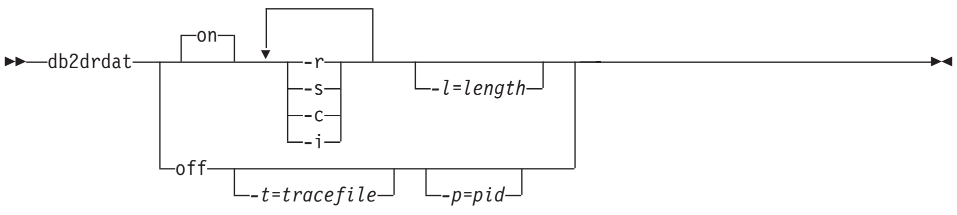
## db2drdat - DRDA Trace

Allows the user to capture the DRDA data stream exchanged between a DRDA Application Requestor (AR) and the DB2 UDB DRDA Application Server (AS). Although this tool is most often used for problem determination, by determining how many sends and receives are required to execute an application, it can also be used for performance tuning in a client/server environment.

### Authorization

None

### Command Syntax



### Command Parameters

**on** Turns on AS trace events (all if none specified).

**off** Turns off AS trace events.

**-r** Traces DRDA requests received from the DRDA AR.

**-s** Traces DRDA replies sent to the DRDA AR.

**-c** Traces the SQLCA received from the DRDA server on the host system. This is a formatted, easy-to-read version of *not null* SQLCAs.

**-i** Includes time stamps in the trace information.

**-l** Specifies the size of the buffer used to store the trace information.

**-p** Traces events only for this process. If -p is not specified, all agents with incoming DRDA connections on the server are traced.

**Note:** The *pid* to be traced can be found in the *agent* field returned by "LIST APPLICATIONS" on page 295.

**-t** Specifies the destination for the trace. If a file name is specified without a complete path, missing information is taken from the current path.

**Note:** If *tracefile* is not specified, messages are directed to `db2drdat.dmp` in the current directory.

## Usage Notes

Do not issue **db2trc** commands while **db2drdat** is active (for information about the **db2trc** command, see the *Troubleshooting Guide*).

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

1. -r
   - Type of DRDA request
   - Receive buffer.
2. -s
   - Type of DRDA reply / object
   - Send buffer.
3. CPI-C error information
   - Severity
   - Protocol used
   - API used
   - Local LU name
   - Failed CPI-C function
   - CPI-C return code.

The command returns an exit code. A zero value indicates that the command completed successfully, and a nonzero value indicates that the command was not successful.

**Note:** If **db2drdat** sends the output to a file that already exists, the old file will be erased unless the permissions on the file do not allow it to be erased, in which case the operating system will return an error.

## db2empfa - Enable Multi-page File Allocation

Enables multi-page file allocation for a database.

### Scope

This command only affects the node on which it is executed.

### Authorization

*sysadm*

### Required Connection

None. This command establishes a database connection.

### Command Syntax

```
►►──db2empfa──database-alias──────────────────────────────────────────►◄
```

### Command Parameters

**database-alias**
Specifies the alias of the database for which multi-page file allocation is to be enabled.

### Usage Notes

This utility:

- Connects to the database partition on a node (where applicable) in exclusive mode
- In all SMS table spaces, allocates empty pages to fill up the last extent in all data and index files which are larger than one extent
- Changes the value of the database configuration parameter *multipage_alloc* to `YES`
- Disconnects.

Since **db2empfa** connects to the database partition on a node in exclusive mode, it cannot be run concurrently on the catalog node, or on any other node.

## db2eva - Event Analyzer

Starts the event analyzer, allowing the user to trace performance data produced by DB2 event monitors that have their data directed to files. See the *System Monitor Guide and Reference* for more information on event monitors.

### Authorization

None, unless connecting to the database and selecting from the catalogs (`-evm`, `-db`, and `-conn`); then one of the following is required:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

### Required Connection

None

### Command Syntax

```
►►──db2eva───┬─-path──evmon-target──┬────────────────────────────┬─┬───────────►◄
             │                      └─-conn──┬─────────────────┬─┘ │
             │                               └─-db──-database-alias─┘ │
             └─-evm──evmon-name──-db──database-alias──┬───────┬──────┘
                                                      └─-conn─┘
```

### Command Parameters

**-path evmon-target**
Specifies the directory containing the event monitor trace files.

**-conn**
Requests that **db2eva** maintain a connection to the database specified with `-db`, or if `-db` is not used, then to the database specified in the event monitor trace header. Maintaining a connection allows the event analyzer to obtain information not contained in the trace files (for example, the text for static SQL). A statement event record contains the package creator, package, and section number; when `-conn` is specified, **db2eva** can retrieve the text from the database system catalog (`sysibm.sysstmt`).

**-db database-alias**
Specifies the name of the database defined for the event monitor. If `-path` is specified, the database name in the event monitor trace header is overridden.

**-evm evmon-name**
Specifies the name of the event monitor whose traces are to be analyzed.

## Usage Notes

Although there is no required connection, **db2eva** will attempt to connect to the database if the `-conn`, or the `-evm` and the `-db` options are used. If the user can access the database and has the appropriate authorization, the SQL text for static statements can be displayed. Without the required access or authority, only the text for dynamic statements is available.

There are two methods for reading event monitor traces:

1. Specifying the directory where the trace files are located (using the `-path` option). This allows users to move trace files from a server and analyze them locally. This can be done even if the event monitor has been dropped.
2. Specifying the database and event monitor names allows automatic location of the trace files. The event analyzer connects to the database, and issues a `select target from sysibm.syseventmonitors` to locate the directory where the event monitor writes its trace files. The connection is then released, unless `-conn` was specified. This method cannot be used if the event monitor has been dropped.

**Note:** The event analyzer can be used to analyze the data produced by an active event monitor. However, event monitors buffer their data before writing it to disk; therefore, some information may be missing. Turn off the event monitor, thereby forcing it to flush its buffers.

## db2evmon - Event Monitor Productivity Tool

Formats event monitor file and named pipe output, and writes it to standard output.

**Note:** This productivity tool is provided *as is*, without any warranty of any kind, including the warranties of merchantability and fitness for a particular purpose, which are expressly disclaimed.

### Authorization

None, unless connecting to the database (`-evm`, `-db`,); then, one of the following is required:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

### Required Connection

None

### Command Syntax

```
►►──db2evmon──┬─┬───────────────────────────────────────────────┬─┬──────────►◄
              │ └─-db──database-alias───-evm──event-monitor-name─┘ │
              └─-path──event-monitor-target────────────────────────┘
```

### Command Parameters

**-db database-alias**
: Specifies the database whose data is to be displayed. This parameter is case sensitive.

**-evm event-monitor-name**
: The one-part name of the event monitor. An ordinary or delimited SQL identifier. This parameter is case sensitive.

**-path event-monitor-target**
: Specifies the directory containing the event monitor trace files.

### Usage Notes

If the data is being written to files, the tool formats the files for display using standard output. In this case, the monitor is turned on first, and any event data in the files is displayed by the tool. To view any data written to files after the tool has been run, reissue **db2evmon**.

If the data is being written to a pipe, the tool formats the output for display using standard output as events occur. In this case, the tool is started *before* the monitor is turned on.

## db2exfmt - Explain Table Format Tool

Formats the contents of the explain tables.

For a complete description of this command, see the *Administration Guide*.

### See Also

"db2expln - DB2 SQL Explain Tool" on page 39.

## db2expln - DB2 SQL Explain Tool

Describes the access plan selection for static SQL statements in packages that are stored in the DB2 common server system catalogs. Given a database name, package name, package creator, and section number, the tool interprets and describes the information in these catalogs.

For a complete description of this command, see the *Administration Guide*.

### See Also

"db2exfmt - Explain Table Format Tool" on page 38.

## db2flsn - Find Log Sequence Number

Returns the name of the file that contains the log record identified by a specified log sequence number (LSN).

### Authorization

None

### Command Syntax

```
►►──db2flsn──┬──────┬──input_LSN──────────────────────────────────────────►◄
             └─-q───┘
```

### Command Parameters

**-q** Specifies that only the log file name be printed. No error or warning messages will be printed, and status can only be determined through the return code. Valid error codes are:

- -100 Invalid input
- -101 Cannot open LFH file
- -102 Failed to read LFH file
- -103 Invalid LFH
- -104 Database is not recoverable
- -105 LSN too big
- -500 Logical error.

Other valid return codes are:

- 0 Successful execution
- 99 Warning: the result is based on the last known log file size.

**input_LSN**
A 12-byte string that represents the internal (6-byte) hexadecimal value with leading zeros.

### Examples

```
db2flsn 000000BF0030
   Given LSN is contained in log file S0000002.LOG

db2flsn -q 000000BF0030
   S0000002.LOG

db2flsn 000000BE0030
   Warning: the result is based on the last known log file size.
   The last known log file size is 23 4K pages starting from log extent 2.
   Given LSN is contained in log file S0000001.LOG

db2flsn -q 000000BE0030
   S0000001.LOG
```

## Usage Notes

The log header control file `sqlogctl.lfh` must reside in the current directory. Since this file is located in the database directory, the tool can be run from the database directory, or the control file can be copied to the directory from which the tool will be run.

The tool uses the *logfilsiz* database configuration parameter. DB2 records the three most recent values for this parameter, and the first log file that is created with each *logfilsiz* value; this enables the tool to work correctly when *logfilsiz* changes. If the specified LSN predates the earliest recorded value of *logfilsiz*, the tool uses this value, and returns a warning. The tool can be used with database managers prior to UDB Version 5.2; in this case, the warning is returned even with a correct result (obtained if the value of *logfilsiz* remains unchanged).

This tool can only be used with recoverable databases. A database is recoverable if it is configured with *logretain* or *userexit* on.

## See Also

“sqlurlog - Asynchronous Read Log” API in the *Administrative API Reference*.

## db2gov - DB2 Governor

Monitors and changes the behavior of applications that run against a database. By default, a daemon is started on every logical node, but the front-end utility can be used to start a single daemon at a specific node to monitor the activity against the database partition at that node.

Each governor daemon collects statistics about the applications running against a database. It then checks these statistics against the rules that were specified in the governor configuration file that applies to that specific database. The governor then acts according to these rules. For example, a rule may indicate that an application is using too much resource. In this situation, the governor may change the application’s priority or force it off the database, according to instructions specified in the governor configuration file.

For a complete description of this command, see the *Administration Guide*.

### See Also

“db2govlg - DB2 Governor Log Query” on page 43.

## db2govlg - DB2 Governor Log Query

Extracts records of specified type from the governor log files (see “db2gov - DB2 Governor” on page 42). The DB2 governor monitors and changes the behavior of applications that run against a database.

### Authorization

None

### Command Syntax

```
►►──db2govlg──log-file──┬────────────────────┬──┬─────────────────────────┬──────────────►◄
                        └─nodenum──node-num──┘  └─rectype──record-type──┘
```

### Command Parameters

**log-file**
The base name of one or more log files that are to be queried.

**nodenum node-num**
Number of the node on which the governor is running.

**rectype record-type**
The type of record that is to be queried. Valid record types are:

- START
- FORCE
- NICE
- ERROR
- WARNING
- READCFG
- STOP
- ACCOUNT

### See Also

“db2gov - DB2 Governor” on page 42.

## db2icrt - Create Instance

Creates DB2 instances.

On UNIX based systems, this utility is located in the `DB2DIR/instance` directory, where `DB2DIR` represents `/usr/lpp/db2_05_00` on AIX, and `/opt/IBMdb2/V5.0` on both Solaris and HP-UX. On OS/2 or the Windows operating system, it is located in the `\sqllib\bin` subdirectory.

For a complete description of this command, see one of the *Quick Beginnings* books.

## db2idrop - Remove Instance

Removes a DB2 instance that was created by “db2icrt - Create Instance” on page 44. Removes the instance entry from the list of instances.

On UNIX based systems, this utility is located in the `DB2DIR/instance` directory, where `DB2DIR` represents `/usr/lpp/db2_05_00` on AIX, and `/opt/IBMdb2/V5.0` on both Solaris and HP-UX. On OS/2 or the Windows operating system, it is located in the `\sqllib\bin` subdirectory.

For a complete description of this command, see the *Administration Guide*.

## db2ilist - List Instances

Lists all the instances that are available on a system.

On UNIX based systems, this utility is located in the `DB2DIR/bin` directory, where `DB2DIR` represents `/usr/lpp/db2_05_00` on AIX, and `/opt/IBMdb2/V5.0` on both Solaris and HP-UX. On OS/2 or the Windows operating system, it is located in the `\sqllib\bin` subdirectory.

For a complete description of this command, see the *Administration Guide*.

## db2imigr - Migrate Instance

Migrates an existing instance following installation of the database manager.

On UNIX based systems, this utility is located in the `DB2DIR/instance` directory, where `DB2DIR` represents `/usr/lpp/db2_05_00` on AIX, and `/opt/IBMdb2/V5.0` on both Solaris and HP-UX. On OS/2 or the Windows operating system, it is located in the `\sqllib\bin` subdirectory.

For a complete description of this command, see one of the *Quick Beginnings* books.

## db2ipxad - Get IPX/SPX Internetwork Address

Returns the DB2 server’s IPX/SPX internetwork address. This command *must* be issued locally from the DB2 server machine. Issuing the command from a remote client is not supported. The internetwork address can be used on a client machine to catalog an IPX/SPX node using ″direct addressing″. For more information, see one of the *Quick Beginnings* books.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──db2ipxad──────────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### See Also

“CATALOG IPX/SPX NODE” on page 162.

## db2iupdt - Update Instances

Updates a specified DB2 instance to enable acquisition of a new system configuration or access to function associated with the installation or removal of certain product options.

On UNIX based systems, this utility is located in the `DB2DIR/instance` directory, where `DB2DIR` represents `/usr/lpp/db2_05_00` on AIX, and `/opt/IBMdb2/V5.0` on both Solaris and HP-UX. On OS/2 or the Windows operating system, it is located in the `\sqllib\bin` subdirectory.

For a complete description of this command, see the *Administration Guide*.

## db2licm - License Management Tool

Performs basic license functions in the absence of the Control Center. Adds, removes, lists, and modifies licenses installed on the local system.

### Authorization

On UNIX based systems, root authority is required. On OS/2 or the Windows operating system, no authorization is required.

### Required Connection

None

### Command Syntax

```
►►──db2licm──┬─────────────────────────────────────────────────┬──────────►◄
             ├─-a──filename────────────────────────────────────┤
             ├─-l──────────────────────────────────────────────┤
             ├─-p──prod-password──┬────────────┬──┬────────────┬┤
             │                    └─REGISTERED─┘  └─CONCURRENT─┘│
             ├─-r──prod-password───────────────────────────────┤
             ├─-u──prod-password──num-users────────────────────┤
             ├─-e──┬─HARD─┬────────────────────────────────────┤
             │     └─SOFT─┘                                    │
             ├─-v──────────────────────────────────────────────┤
             └─┬─-h─┬──────────────────────────────────────────┘
               └─-?─┘
```

### Command Parameters

**-a filename**
: Adds a license for a product. Specify a file name containing valid license information.

**-l**
: Lists all the products with available license information.

**-p prod-password REGISTERED CONCURRENT**
: Updates the license policy type to use on the system. REGISTERED, CONCURRENT, or both keywords can be specified.

**-r prod-password**
: Removes the license for a product. After the license is removed, the product functions in ″Try & Buy″ mode. To get the password for a specific product, invoke the command with the **-l** option.

**-u prod-password num-users**
: Updates the number of user licenses that the customer has purchased. Specify the number of users, and the password of the product for which the licenses were purchased.

**-e** Updates the enforcement policy on the system. Valid values are: HARD and SOFT. HARD specifies that unlicensed requests will not be allowed. SOFT specifies that unlicense requests will be logged but not restricted.

**-v** Displays version information.

**-h/-?** Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## Examples

```
db2licm -a db2entr.lic
db2licm -p ubca874koldji93ldf registered concurrent
db2licm -r abce874koldji93ldf
db2licm -u 10 abce874koldji93Idf
```

## db2look - DB2 Statistics Extraction Tool

Generates the update statements required to make the catalog statistics of a test database match those of a production database.

It is often advantageous to have a test system contain a subset of the production system's data. However, access plans selected for such a test system are not necessarily the same as those that would be selected for the production system. Both the catalog statistics and the configuration parameters for the test system must be updated to match those of the production system. This tool queries the system catalogs of a database, and outputs table space, table, index, and column information about each table in that database.

### Authorization

SELECT privilege on the system catalogs.

### Required Connection

None. This command establishes a database connection.

### Command Syntax

```
►►──db2look──-d──DBname──┬──────────────┬─┬────────────────────────────────┬──────────►
                         └─-u──Creator──┘ └─┬────┬─┬────┬─┬────┬─-t──Tname─┘
                                            └─-s─┘ └─-g─┘ └─-a─┘

►─┬────────────────────────────────────────────────────────────────────┬──────────────►◄
  └─┬────┬─-o──Fname─┬────┬─┬────┬─┬────┬─┬────┬─┬────┬────────────────┘
    └─-p─┘           └─-e─┘ └─-m─┘ └─-c─┘ └─-r─┘ └─-h─┘
```

### Command Parameters

**-d DBname**
Alias name of the production database that is to be queried.

**-u Creator**
Creator ID. If option `-a` is specified, this parameter is ignored. If neither `-u` nor `-a` is specified, the environment variable **USER** is used.

**-s** Generate a PostScript file.

**Notes:**

1. This option removes all LaTeX and `.tmp` PostScript files.
2. Required non-IBM software: LaTeX, dvips.
3. The `psfig.tex` file must be in the LaTeX input path.

**-g** Use a graph to show fetch page pairs for indices.

**Notes:**

1. This option generates a *filename*.`ps` file, as well as the LaTeX file.
2. Required non-IBM software: Gnuplot.
3. The `psfig.tex` file must be in the LaTeX input path.

**-a** Generate statistics for all users on the database. If used with option `-e`, it generates the DDL for all user tables in the database, but statistics are not generated.

**Notes:**

1. Some DDL characteristics are not extracted by **db2look**.
2. If neither `-u` nor `-a` is specified, the environment variable **USER** is used. On UNIX based systems, this variable does not have to be explicitly set; on Windows NT, however, there is no default value for the **USER** environment variable: on this platform, a user variable in the SYSTEM variables must be set, or a `set USER=<username>` must be issued for the session.

**-t Tname**
Table name. Limits the output to a particular table.

**-p** Use plain text format.

**-o Fname**
If using LaTeX format, write the output to *filename*.`tex`. If using plain text format, write the output to *filename*.`txt`. If this option is not specified, output is written to standard output.

**-e** Extract DDL statements to recreate database data objects. This option generates a CLP script containing DDL statements to recreate tables and indexes. The CLP script can then be run against another database to recreate the database. This option can be used in conjunction with the `-m` option. It does not currently support:

- Creation of triggers
- Creation of UDFs
- Reference to UDFs.

**-m** Run the program in *mimic* mode. This option generates a CLP script containing all the UPDATE statements required to capture the catalog statistics of the production database. The CLP script can then be run against a smaller test database, and the updated test database can be used to validate access plans for production. The `-p`, `-g`, and `-s` options are ignored in mimic mode.

**-c** Do not generate COMMIT statements in mimic mode. The default action is to generate COMMIT statements. In addition, do not generate CONNECT or CONNECT RESET statements. If option `-m` is not specified, this option is ignored.

**-r**
Do not include the RUNSTATS command in mimic mode. The default action is to issue the RUNSTATS command. If option `-m` is not specified, this option is ignored.

**-h**
Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## Examples

Write the statistics for tables created by USER in database DEPARTMENT in LaTeX format to file `output.tex`:

```
db2look -d department -o output
```

Write the statistics for all tables in database DEPARTMENT in LaTeX format to file `output.tex`:

```
db2look -a -d department -o output
```

Write the statistics for all tables in database DEPARTMENT in plain text format to file `output.txt`:

```
db2look -p -a -d department -o output
```

Write the statistics for all tables in database DEPARTMENT for user JAMES. Use a graph to show page fetch pairs. Save the LaTeX output in file `output.tex`. Save the PostScript output in file `output.ps`:

```
db2look -g -s -u james -d department -o output
```

Write the replica CLP commands for table EMPLOYEE in database PAYROLL created by anyone to file `employee.mimic`:

```
db2look -m -a -d payroll -t employee -o employee.mimic
```

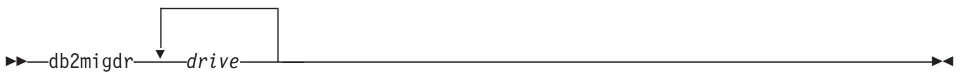
## db2migdr - Local Database Directory Migration

Migrates a down-level local database directory to the current release format, so that databases stored in the local database directory can be accessed for migration.

### Authorization

*sysadm*

### Command Syntax



### Command Parameters

**drive** Specifies the drive where the local database directory can be found. The local database directory must be named `sqldbdir`, and it must be the first level subdirectory off the drive that is specified. The following example invokes the command to migrate the local database directory from drive `c` and from drive `f`:

```
db2migdr c f
```

## db2move - Database Movement Tool

This tool facilitates the movement of large numbers of tables between DB2 databases located on workstations. The tool queries the system catalog tables for a particular database and compiles a list of all user tables. It then exports these tables in PC/IXF format. The PC/IXF files can be imported or loaded to another local DB2 database on the same system, or can be transferred to another workstation platform and imported or loaded to a DB2 database on that platform.

### Authorization

This tool calls the DB2 export, import, and load APIs, depending on the action requested by the user. Therefore, the requesting user ID must have the correct authorization required by those APIs, or the request will fail.

### Command Syntax

```
                                   ┌─────────────┐
                                   ▼             │
►►──db2move──dbname──action────────┬─────────┬───┴──────────────────────────────────►◄
                                   ├──-tc────┤
                                   ├──-tn────┤
                                   ├──-io────┤
                                   ├──-lo────┤
                                   ├──-l─────┤
                                   ├──-u─────┤
                                   └──-p─────┘
```

### Command Parameters

**dbname**
: Name of the database.

**action**
: Must be one of: EXPORT, IMPORT, or LOAD.

**-tc**
: table-creators. The default is all creators.

This is an EXPORT action only. If specified, only those tables created by the creators listed with this option are exported. If not specified, the default is to use all creators. When specifying multiple creators, each must be separated by commas; no blanks are allowed between creator IDs. The maximum number of creators that can be specified is 10. This option can be used with the “-tn” table-names option to select the tables for export.

An asterisk (*) can be used as a wildcard character that can be placed anywhere in the string.

**-tn**
: table-names. The default is all user tables.

This is an EXPORT action only. If specified, only those tables whose names match exactly those in the specified string are exported. If not specified, the default is to use all user tables. When specifying multiple table names, each must be separated by commas; no blanks are allowed between table names. The maximum number of table names that can be specified is 10. This option can be used with the "-tc" table-creators option to select the tables for export. **db2move** will only export those tables whose names are matched with specified table names and whose creators are matched with specified table creators.

An asterisk (*) can be used as a wildcard character that can be placed anywhere in the string.

**-io** import-option. The default is REPLACE_CREATE.

Valid options are: INSERT, INSERT_UPDATE, REPLACE, CREATE, and REPLACE_CREATE.

**-lo** load-option. The default is INSERT.

Valid options are: INSERT and REPLACE.

**-l** lobpaths. The default is the current directory.

This option specifies the absolute path names where LOB files are created (as part of EXPORT) or searched for (as part of IMPORT or LOAD). When specifying multiple LOB paths, each must be separated by commas; no blanks are allowed between LOB paths. If the first path runs out of space (during EXPORT), or the files are not found in the path (during IMPORT or LOAD), the second path will be used, and so on.

If the action is EXPORT, and LOB paths are specified, all files in the LOB path directories are deleted, the directories are removed, and new directories are created. If not specified, the current directory is used for the LOB path.

**-u** userid. The default is the logged on user ID.

Both user ID and password are optional. However, if one is specified, the other must be specified. If the command is run on a client connecting to a remote server, user ID and password should be specified.

**-p** password. The default is the logged on password.

Both user ID and password are optional. However, if one is specified, the other must be specified. If the command is run on a client connecting to a remote server, user ID and password should be specified.

## Examples

- `db2move sample export`

  This will export all tables in the SAMPLE database; default values are used for all options.
- `db2move sample export -tc userid1,us*rid2 -tn tbname1,*tbname2`

  This will export all tables created by “userid1” or user IDs LIKE “us%rid2”, and with the name “tbname1” or table names LIKE “%tbname2”.
- `db2move sample import -l D:\LOBPATH1,C:\LOBPATH2`

  This example is applicable to OS/2 or the Windows operating system only. The command will import all tables in the SAMPLE database; LOB paths “D:\LOBPATH1” and “C:\LOBPATH2” are to be searched for LOB files.
- `db2move sample load -l /home/userid/lobpath,/tmp`

  This example is applicable to UNIX based systems only. The command will load all tables in the SAMPLE database; both the `/home/userid/lobpath` subdirectory and the `tmp` subdirectory are to be searched for LOB files.
- `db2move sample import -io replace -u userid -p password`

  This will import all tables in the SAMPLE database in REPLACE mode; the specified user ID and password will be used.

## Usage Notes

This tool exports, imports, or loads user-created tables. If a database is to be duplicated from one operating system to another operating system, **db2move** facilitates the movement of the tables. It is also necessary to move all other objects associated with the tables, such as: aliases, views, triggers, user-defined functions, and so on. “db2look - DB2 Statistics Extraction Tool” on page 52 can facilitate the movement of some of these objects by extracting the data definition language (DDL) statements from the database.

When export, import, or load APIs are called by **db2move**, the `FileTypeMod` parameter is set to `lobsinfile`. That is, LOB data is kept in separate files from PC/IXF files. There are 26 000 file names available for LOB files.

The LOAD action must be run locally on the machine where the database and the data file reside. When the load API is called by **db2move**, the `CopyTargetList` parameter is set to NULL; that is, no copying is done. If *logretain* is on, the load operation cannot be rolled forward later. The table space where the loaded tables reside is placed in backup pending state, and is not accessible. A full database backup, or a table space backup, is required to take the table space out of backup pending state.

**Files Required/Generated When Using EXPORT:**

- Input: None.

- Output:

| | |
|---|---|
| **EXPORT.out** | The summarized result of the EXPORT action. |
| **db2move.lst** | The list of original table names, their corresponding PC/IXF file names (tabnnn.ixf), and message file names (tabnnn.msg). This list, the exported PC/IXF files, and LOB files (tabnnnc.yyy) are used as input to the **db2move** IMPORT or LOAD action. |
| **tabnnn.ixf** | The exported PC/IXF file of a specific table. |
| **tabnnn.msg** | The export message file of the corresponding table. |
| **tabnnnc.yyy** | The exported LOB files of a specific table.<br><br>“nnn” is the table number. “c” is a letter of the alphabet. “yyy” is a number ranging from 001 to 999.<br><br>These files are created only if the table being exported contains LOB data. If created, these LOB files are placed in the “lobpath” directories. There are a total of 26 000 possible names for the LOB files. |
| **system.msg** | The message file containing system messages for creating or deleting file or directory commands. This is only used if the action is EXPORT, and a LOB path is specified. |

**Files Required/Generated When Using IMPORT:**

- Input:

| | |
|---|---|
| **db2move.lst** | An output file from the EXPORT action. |
| **tabnnn.ixf** | An output file from the EXPORT action. |
| **tabnnnc.yyy** | An output file from the EXPORT action. |

- Output:

| | |
|---|---|
| **IMPORT.out** | The summarized result of the IMPORT action. |
| **tabnnn.msg** | The import message file of the corresponding table. |

**Files Required/Generated When Using LOAD:**

- Input:

| | |
|---|---|
| **db2move.lst** | An output file from the EXPORT action. |
| **tabnnn.ixf** | An output file from the EXPORT action. |
| **tabnnnc.yyy** | An output file from the EXPORT action. |

- Output:

| | |
|---|---|
| **LOAD.out** | The summarized result of the LOAD action. |

| | |
|---|---|
| **tabnnn.msg** | The LOAD message file of the corresponding table. |

## db2mscs - Set up Windows NT Failover Utility

Creates the infrastructure for DB2 to support failover on the Windows NT environment using MSCS support. This utility can be used to enable failover in both single-partition and partitioned database environments.

Run the utility once for each instance on its instance-owning machine. If there is only one DB2 instance running on one machine in the MSCS cluster, a hot-standby configuration is established. If an instance is running on each machine in the MSCS cluster, run the utility once on each instance-owing machine to set up a mutual takeover configuration.

This utility performs the following steps:

1. Reads the required MSCS and DB2 parameters from an input file called `DB2MSCS.CFG`. For information about the full set of input parameters, see the *Administration Guide*.
2. Validates the parameters in the input file.
3. Registers the DB2 resource type.
4. Creates the MSCS group (or groups) to contain the MSCS and DB2 resources.
5. Creates the IP resource.
6. Creates the Network Name resource.
7. Moves MSCS disks to the group.
8. Creates the DB2 resource (or resources).
9. Adds all required dependencies for the DB2 resource.
10. Converts the non-clustered DB2 instance into a clustered instance.
11. Brings all resources online.

For a complete description of this command, see the *Administration Guide*.

## db2perfc - Reset Database Performance Values

Used with the Windows NT Performance Monitor. Resets the performance values for one or more databases. When an application calls the DB2 monitor APIs, the information returned is normally the cumulative values since the DB2 server was started. However, it is often useful to reset performance values, run a test, reset the values again, and then rerun the test.

For a complete description of this command, see the *Administration Guide, Design and Implementation*.

## db2perfi - Performance Counters Registration Utility

Registers the DLL for the DB2 for Windows NT Performance Counters. This must be done to make DB2 and DB2 Connect performance information accessible to the Windows NT Performance Monitor. It also enables any other Windows NT application using the Win32 performance APIs to get performance data.

For a complete description of this command, see the *Administration Guide, Design and Implementation*.

## db2perfr - Performance Monitor Registration Tool

Used with the Windows NT Performance Monitor. Registers an administrator user name and password with DB2. This must be done before Windows NT performance objects can be seen from another DB2 for Windows NT machine. (The default Windows NT Performance Monitor username, SYSTEM, is a DB2 reserved word and cannot be used.)

For a complete description of this command, see the *Administration Guide, Design and Implementation*.

## db2profc - DB2 SQLJ Profile Customizer

Processes an SQLJ profile containing embedded SQL statements. By default, a DB2 package is created in the database; this utility augments the profile with DB2-specific information for use at run time. This utility should be run after the SQLJ application has been translated, but before the application is run.

### Authorization

One of the following:

- *sysadm* or *dbadm* authority
- BINDADD privilege if a package does not exist, and one of:
  - IMPLICIT_SCHEMA authority on the database if the schema name of the package does not exist
  - CREATEIN privilege on the schema if the schema name of the package exists
- ALTERIN privilege on the schema if the package exists
- BIND privilege on the package if it exists.

The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements. If the user has *sysadm* authority, but not explicit privileges to complete the bind, the database manager grants explicit *dbadm* authority automatically.

### Required Connection

This command establishes a database connection.

### Command Syntax

```
►►──db2profc──┬────────────────────────────────────────────┬──────────────►
              └─-user=──username──-password=──password──┘

►─┬──────────────────────────────────────┬─-url=──JDBC-url──profilename──────►◄
  └─-prepoptions=──"precompile-options"──┘
```

### Command Parameters

**-user= username**
: Specifies the name used when connecting to a database to perform profile customization.

**-password= password**
: Specifies the password for the user name.

**-prepoptions= ″precompile-options″**
: Specifies a list of precompile options to be used by the DB2 precompiler. For a list of supported precompile options, see the *Application Development Guide*.

  The precompile option ″PACKAGE USING package-name″ specifies the name of the package that is to be generated by the precompiler. If a name is not entered, the name of the profile (minus extension and folded to uppercase) is used. Maximum length is 8 characters.

  The precompile option ″BINDFILE USING bind-file″ specifies the name of the bind file that is to be generated by the precompiler. The file name must have an extension of `.bnd`. If a file name is not entered, the precompiler uses the name of the profile, and adds the `.bnd` extension. If a path is not provided, the bind file is created in the current directory.

**-url= JDBC-url**
: Specifies a JDBC URL for establishing the database connection.

**profilename**
: Specifies the name of a profile in which SQL statements are stored. When an SQLJ file is translated into a Java file, information about the SQL operations it contains is stored in SQLJ-generated resource files called profiles. Profiles are identified by the suffix `_SJProfile`*N* (where *N* is an integer) following the name of the original input file. They have a `.ser` extension. Profile names can be specified with or without the `.ser` extension.

## Examples

```
db2profc -user=username -password=password -url=JDBC-url
   -prepoptions="bindfile using pgmname1.bnd package using pgmname1"
   pgmname_SJProfile1.ser
```

## Usage Notes

For more information about SQLJ, see the *Application Development Guide*.

## See Also

"db2profp - DB2 SQLJ Profile Printer" on page 67.

## db2profp - DB2 SQLJ Profile Printer

Prints the contents of a DB2 customized version of a profile in plain text.

### Authorization

None

### Required Connection

This command establishes a database connection.

### Command Syntax

```
>>──db2profp──┬──────────────────────────────────────────┬──-url=──JDBC-url──────────►
              └─-user=──username──-password=──password───┘

   ┌──────────────┐
   ▼              │
►───profilename───┴──────────────────────────────────────────────────────────────────►◄
```

### Command Parameters

**-user= username**
: Specifies the name used when connecting to a database to print the customized profile.

**-password= password**
: Specifies the password for the user name.

**-url= JDBC-url**
: Specifies a JDBC URL for establishing the database connection.

**profilename**
: Specifies one or more profiles in which SQL statements are stored. When an SQLJ file is translated into a Java file, information about the SQL operations it contains is stored in SQLJ-generated resource files called profiles. Profiles are identified by the suffix `_SJProfile`*N* (where *N* is an integer) following the name of the original input file. They have a `.ser` extension. Profile names can be specified with or without the `.ser` extension.

### Examples

```
db2profp -user=username -password=password -url=JDBC-url
   pgmname_SJProfile1.ser
```

### Usage Notes

For more information about SQLJ, see the *Application Development Guide*.

### See Also

"db2profc - DB2 SQLJ Profile Customizer" on page 65.

## db2rbind - Rebind all Packages

Rebinds packages in a database.

### Authorization

One of the following:

- *sysadm*
- *dbadm*

### Required Connection

None

### Command Syntax

```
►►──db2rbind──database──/l logfile──┬─────┬──┬──────────────────────────┬──────────►◄
                                    └─all─┘  └─/u userid──/p password──┘
```

### Command Parameters

**database**
Specifies an alias name for the database whose packages are to be revalidated.

**/l**
Specifies the (optional) path and the (mandatory) file name to be used for recording errors that result from the package revalidation procedure.

**all**
Specifies that rebinding of all valid and invalid packages is to be done. If this option is not specified, all packages in the database are examined, but only those packages that are marked as invalid are rebound, so that they are not rebound implicitly during application execution.

**/u**
User ID. This parameter must be specified if a password is specified.

**/p**
Password. This parameter must be specified if a user ID is specified.

### Usage Notes

This command uses the CLP REBIND command to attempt the revalidation of all packages in a database. Use of **db2rbind** is not mandatory. One may choose to allow package revalidation to occur implicitly when the package is first used, or to selectively revalidate particular packages with either the REBIND or the BIND command.

## See Also

"BIND" on page 131

"REBIND" on page 403.

## db2sampl - Create Sample Database

Creates a sample database named SAMPLE. For more information about this database, see the *SQL Reference*.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Command Syntax

```
►►─db2sampl──┬──────┬──┬────┬─────────────────────────────────────────────►◄
             └─path─┘  └─-k─┘
```

### Command Parameters

**path** Specifies the path on which to create the SAMPLE database. The path is a single drive letter for OS/2 and Windows.

If a path is not specified, SAMPLE is created on the default database path (the *dftdbpath* parameter in the database manager configuration file). On UNIX based systems, the default is the HOME directory of the instance owner. On OS/2 or the Windows operating system, it is the root directory (where DB2 is installed).

**-k** Creates primary keys on the following SAMPLE tables:

| Table | Primary Key |
|---|---|
| DEPARTMENT | DEPTNO |
| EMPLOYEE | EMPNO |
| ORG | DEPTNUMB |
| PROJECT | PROJNO |
| STAFF | ID |
| STAFFG | ID (DBCS only) |

**Note:** The path must be specified *before* this option.

### Usage Notes

This command can only be executed from server nodes. SAMPLE cannot be created on nodes that are database clients only.

The SAMPLE database is created with the instance authentication type that is specified by the database manager configuration parameter *authentication* (see “GET DATABASE MANAGER CONFIGURATION” on page 236).

The qualifiers for the tables in SAMPLE are determined by the user ID issuing the command.

If SAMPLE already exists, **db2sampl** creates the tables for the user ID issuing the command, and grants the appropriate privileges.

## db2set - DB2 Profile Registry Command

Displays, sets, or removes DB2 profile variables. An external environment registry command that supports local and remote administration, via the DB2 Administration Server, of DB2's environment variables stored in the DB2 profile registry.

### Authorization

*sysadm*

### Required Connection

None

### Command Syntax

```
►►──db2set──┬──────────────────────┬──┬────────────────────────────────────┬───────────►
            └─variable=──value─────┘  ├─-g─────────────────────────────────┤
                                      └──┬─────────────────────────────────┤
                                         └─-i──instance──┬───────────────┬─┘
                                                         └─node-number───┘

►──┬───────┬──┬────────┬──┬─────────────────────────────┬───────────────────────────►
   └─-all──┘  └─-null──┘  └─-r──instance──┬───────────────┬┘
                                          └─node-number───┘

►──┬─────────────────────────────────────────────────────┬──┬──────┬──┬─────┬──┬─────┬──────►◄
   └─-n──DAS node──┬─────────────────────────────────────┬┘  ├─-l───┤  └─-v──┘  ├─-h──┤
                   └─-u──user──┬─────────────────────────┬┘  └─-lr──┘           └─-?──┘
                               └─-p──password────────────┘
```

### Command Parameters

**variable= value**
: Sets a specified variable to a specified value. To delete a variable, do not specify a value for the specified variable. Changes to settings take effect after the instance has been restarted.

**-g**
: Accesses the global profile variables.

**-i**
: Specifies the instance profile to use instead of the current, or default.

**node-number**
: Specifies a number listed in the `db2nodes.cfg` file.

**-all**
: Displays all occurrences of the local environment variables as defined in:

- The environment, denoted by [e]

- The node level registry, denoted by [n]
- The instance level registry, denoted by [i]
- The global level registry, denoted by [g].

**-null** Sets the value of the variable at the specified registry level to NULL. This avoids having to look up the value in the next registry level, as defined by the search order.

**-r instance**
Resets the profile registry for the given instance.

**-n DAS node**
Specifies the remote DB2 administration server node name.

**-u user**
Specifies the user ID to use for the administration server attachment.

**-p password**
Specifies the password to use for the administration server attachment.

**-l** Lists all instance profiles.

**-lr** Lists all supported registry variables.

**-v** Specifies verbose mode.

**-h/-?** Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## Examples

- Display all defined profiles (DB2 instances):
```
db2set -l
```
- Display all supported registry variables:
```
db2set -lr
```
- Display all defined global variables:
```
db2set -g
```
- Display all defined variables for the current instance:
```
db2set
```
- Display all defined values for the current instance:
```
db2set -all
```
- Display all defined values for DB2COMM for the current instance:
```
db2set -all DB2COMM
```
- Reset all defined variables for the instance INST on node 3:
```
db2set -r -i INST 3
```
- Unset the variable DB2CHKPTR on the remote instance RMTINST through the DAS node RMTDAS using user ID MYID and password MYPASSWD:

```
db2set -i RMTINST -n RMTDAS -u MYID -p MYPASSWD DB2CHKPTR=
```

- Set the variable DB2COMM to be TCPIP,IPXSPX,NETBIOS globally:
  ```
  db2set -g DB2COMM=TCPIP,IPXSPX,NETBIOS
  ```
- Set the variable DB2COMM to be only TCPIP for instance MYINST:
  ```
  db2set -i MYINST DB2COMM=TCPIP
  ```
- Set the variable DB2COMM to null at the given instance level:
  ```
  db2set -null DB2COMM
  ```

## Usage Notes

If no variable name is specified, the values of all defined variables are displayed. If a variable name *is* specified, only the value of that variable is displayed. To display all the defined values of a variable, specify *variable* `-all`. To display all the defined variables in all registries, specify `-all`.

To modify the value of a variable, specify *variable*=, followed by its new value. To set the value of a variable to NULL, specify *variable* `-null`.

**Note:** Changes to settings take effect after the instance has been restarted.

To delete a variable, specify *variable*=, followed by no value.

## db2setss - DB2 Set Synchronization Session

Displays, sets, or removes the synchronization session identifier for a satellite.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──db2setss──-a=──application-version──┬────┬─────────────────────────────────►◄
                                         ├─-h─┤
                                         └─-?─┘
```

### Command Parameters

**-a= application-version**
Sets the synchronization session identifier for the satellite to the specified application version. To remove the identifier, do not specify a value for the application version. Changes to the session identifier take effect at the next synchronization session.

**-h/-?**
Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

### Examples

Set the application version for the satellite to `V1R0M00`:

```
db2setss -a=V1R0M00
```

### Usage Notes

If only `-a` is specified, the current value of the application version is displayed. To modify the value of the application version, specify `-a=`, followed by its new value. To set the value of the application version to NULL, specify `-a=`, followed by no value.

Changes to the application version take effect at the next synchronization session.

## db2sql92 - SQL92 Compliant SQL Statement Processor

Reads SQL statements from either a flat file or standard input, dynamically describes and prepares the statements, and returns an answer set. Supports concurrent connections to multiple databases.

### Authorization

*sysadm*

### Required Connection

None. This command establishes a database connection.

### Command Syntax

```
>>──db2sql92──┬─────────────┬──┬───────────────┬──┬───────────────────┬──────────────>
              └─-d──dbname──┘  └─-f──file_name─┘  └─-a──userid/passwd─┘

>──┬──────────────────────────────┬──┬──────────────┬──┬───────────────────┬──────────>
   └─-r──outfile──┬─────────────┬─┘  └─-c──┬─on──┬─┘  └─-i──┬─short────┬─┘
                  └─,outfile2───┘          └─off─┘           ├─none─────┤
                                                             ├─long─────┤
                                                             └─complete─┘

>──┬──────────────┬──┬──────────────┬──┬──────────────┬──┬────┬─────────────────────><
   └─-o──options──┘  │      ┌─off─┐ │  │      ┌─off─┐ │  └─-h─┘
                     └─-v───┴─on──┴─┘  └─-s───┴─on──┴─┘
```

### Command Parameters

**-d dbname**
An alias name for the database against which SQL statements are to be applied. The default is the value of the **DB2DBDFT** environment variable.

**-f file_name**
Name of an input file containing SQL statements. The default is standard input.

Identify comment text with two hyphens at the start of each line, that is, `--` <*comment*>. If it is to be included in the output, mark the comment as follows: `--#COMMENT` <*comment*>.

A *block* is a number of SQL statements that are treated as one, that is, information is collected for all of those statements at once, instead of one at a time. Identify the beginning of a block of queries as follows: `--#BGBLK`. Identify the end of a block of queries as follows: `--#EOBLK`.

Specify one or more control options as follows: `--#SET` *`<control option>`* *`<value>`*. Valid control options are:

**ROWS_FETCH**
Number of rows to be fetched from the answer set. Valid values are `-1` to *n*. The default value is `-1` (all rows are to be fetched).

**ROWS_OUT**
Number of fetched rows to be sent to output. Valid values are `-1` to *n*. The default value is `-1` (all fetched rows are to be sent to output).

**AUTOCOMMIT**
Specifies autocommit on or off. Valid values are `ON` or `OFF`. The default value is `ON`.

**PAUSE**
Prompts the user to continue.

**TIMESTAMP**
Generates a time stamp.

**-a userid/passwd**
Name and password used to connect to the database.

**-r outfile**
An output file that will contain the query results. An optional *outfile2* will contain a results summary. The default is standard output.

**-c** Automatically commit changes resulting from each SQL statement.

**-i** An elapsed time interval (in seconds).

**none** Specifies that time information is not to be collected.

**short** The run time for a query.

**long** Elapsed time at the start of the next query.

**complete**
The time to prepare, execute, and fetch, expressed separately.

**-o options**
Control options. Valid options are:

**f rows_fetch**
Number of rows to be fetched from the answer set. Valid values are `-1` to *n*. The default value is `-1` (all rows are to be fetched).

**r rows_out**
Number of fetched rows to be sent to output. Valid values are `-1` to *n*. The default value is `-1` (all fetched rows are to be sent to output).

**-v** Verbose. Send information to standard error during query processing. The default value is `off`.

**-s** Summary Table. Provide a summary of elapsed times and CPU times, containing both the arithmetic and the geometric means of all collected values.

**-h** Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## Usage Notes

The following can be executed from the **db2sql92** command prompt:

- All control options
- SQL statements
- CONNECT statements
- `commit work`
- `help`
- `quit`

This tool supports switching between different databases during a single execution of the program. To do this, issue a CONNECT RESET and then one of the following on the **db2sql92** command prompt (stdin):

```
connect to database
connect to database USER userid USING passwd
```

SQL statements can be up to 32 700 characters in length. Statements must be terminated by a semicolon.

SQL statements are executed with the repeatable read (RR) isolation level.

## See Also

"db2batch - Benchmark Tool" on page 11.

## db2start - Start DB2

Starts the current database manager instance background processes on a single node or on all the nodes defined in a multi-node environment. Start DB2 at the server before connecting to a database, precompiling an application, or binding a package to a database.

**db2start** can be executed as a system command or a CLP command. For a complete description of this command, see "START DATABASE MANAGER" on page 475.

## db2stop - Stop DB2

Stops the current database manager instance.

**db2stop** can be executed as a system command or a CLP command. For a complete description of this command, see “STOP DATABASE MANAGER” on page 480.

## db2sync - Start DB2 Synchronizer

Starts a graphical application that can be used to start, stop, and monitor the progress of a synchronization session for a satellite. This command can be invoked in test mode, wherein the ability of a satellite to synchronize can be validated.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──db2sync──┬────┬──────────────────────────────────────────────►◄
             └─-t─┘
```

### Command Parameters

**-t** Specifies that the synchronizer is to be started in test mode, in which the ability of a satellite to synchronize with its control server can be verified.

### Usage Notes

For general information about satellite synchronization, see the *Administering Satellites Guide and Reference*.

## db2tbst - Get Tablespace State

Accepts a hexadecimal table space state value, and returns the state. The state value is part of the output from “LIST TABLESPACES” on page 332.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──db2tbst──tablespace-state──────────────────────────────────────────►◄
```

### Command Parameters

**tablespace-state**
A hexadecimal table space state value.

### Examples

The request `db2tbst 0x0000` produces the following output:

```
State = Normal
```

### See Also

“LIST TABLESPACES” on page 332.

## db2trc - Trace

Activates DB2 trace facility tracing. DB2 uses its trace facility to trace events, dump trace data to a file, and format trace data into a readable form. Only use the trace facility when directed by DB2 Customer Service or by a technical support representative.

### Authorization

On UNIX based systems, one of the following:

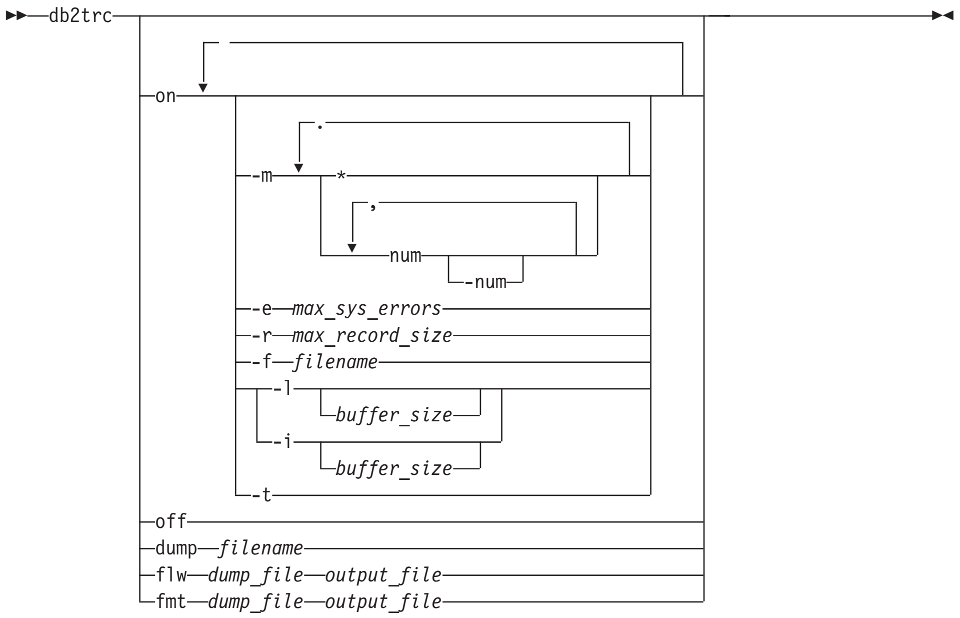
- *sysadm*
- *sysctrl*
- *sysmaint*

On OS/2 or the Windows operating system, no authorization is required.

### Required Connection

None

### Command Syntax



## Command Parameters

**on** Use this parameter to start the DB2 trace facility. See the next section for information about the options for this parameter.

**dump** After reproducing the error, dump the trace information to a file. The following command will put the information in the current directory in a file called `db2trc.dmp`:

```
db2trc dump db2trc.dmp
```

Specify a file name with this parameter. The file is saved in the current directory unless the path is explicitly specified.

**off** After the trace is dumped to a file, turn the trace off by typing:

```
db2trc off
```

**flw | fmt**
After the trace is dumped to a binary file, confirm that it is taken by formatting it into an ASCII file. Use either the flw option (to sort by process or thread), or the fmt option (to list every event chronologically). For either option, specify the name of the dump file and the name of the output file that will be generated. For example:

```
db2trc flw db2trc.dmp db2trc.flw
```

### Starting a DB2 Trace

To use the trace facility, first turn it on using **db2trc on**. Unless instructed otherwise by DB2 Customer Service, use the command without options. The default trace option values are:

- *mask* = * (trace everything)
- *max_sys_errors* = -1 (collect all)
- *max_record_size* = 16KB
- Trace destination = -s (shared memory)
- Records to retain = -l (last)
- *buffer_size* = 64KB if the trace destination is shared memory, or 1MB if the trace destination is a file.

If instructed to specify options to tailor the trace, use the following options as directed by DB2 Customer Service:

**-m** *mask*
Specifies trace record types to focus the search. The *mask* variable consists of four one-byte masks separated by periods. The byte-masks act as a filter to accept or reject the trace record sent by DB2 for each event *based on its ID*. The four one-byte masks correspond to products, event types, components, and functions, respectively.

**-e** *max_sys_errors*
: Limits the number of DB2 internal system errors the trace will hold to *max_sys_errors*.

**-r** *max_record_size*
: Limits the size of trace records to *max_record_size* bytes. Longer trace records are truncated.

**-f** *filename*
: Specifies trace buffer location in shared memory or a file.

**-l [** *buffer_size***] | -i [***buffer_size***]**
: ’-l’ (″l″ as in ″lucky″) specifies that the last trace records are retained (that is, the first records are overwritten when the buffer is full). ’-i’ specifies that the initial trace records are retained (that is, no more records are written to the buffer once it is full). Use either of these options to specify the buffer size.

**-t**
: Includes time stamps. Applicable to UNIX environments only, where the logging of time stamps affects performance.

## Examples

For additional information about **db2trc**, including trace examples, see the *Troubleshooting Guide*.

## Usage Notes

The **db2trc** command must be issued several times to turn tracing on, produce a dump file, format the dump file, and turn tracing off again. The parameter list shows the order in which the parameters should be used.

## db2uiddl - Prepare Unique Index Conversion to V5 Semantics

Facilitates the management of a staged migration of unique indexes on a user's own schedule. Generates CREATE UNIQUE INDEX statements for unique indexes on user tables. For detailed information about using this tool, see one of the *Quick Beginnings* books.

### Authorization

*sysadm*

### Required Connection

Database. This command automatically establishes a connection to the specified database.

### Command Syntax

```
►►──db2uiddl──-d──database-name──┬──────────────────┬──┬────────────────┬──────────►
                                 └─-u──table-schema─┘  └─-t──table-name─┘

►──┬──────────────┬──┬────┬───────────────────────────────────────────────────────►◄
   └─-o──filename─┘  └─-h─┘
```

### Command Parameters

**-d database-name**
The name of the database to be queried.

**-u table-schema**
Specifies the schema (creator user ID) of the tables that are to be processed. The default action is to process tables created by all user IDs.

**-t table-name**
The name of a table that is to be processed. The default action is to process all tables.

**-o filename**
The name of a file to which output is to be written. The default action is to write output to standard output.

**-h** Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

### Usage Notes

This tool can only be used on a database that has been migrated to DB2 Version 5.

**Note:** This tool was not designed to handle certain types of names. If a specific table name or table schema is a delimited identifier containing lower case characters, special characters, or blanks, it is preferable to request processing of *all* tables or schemas. The resulting output can be edited.

## db2untag - Release Container Tag

Removes the DB2 tag on a table space container. The tag is used to prevent DB2 from reusing a container in more than one table space. Displays information about the container tag, identifying the database with which the container is associated. Useful when it is necessary to release a container last used by a database that has since been deleted. If the tag is left behind, DB2 is prevented from using the resource in future.

**Attention:** This tool should only be used by informed system administrators.

### Authorization

The user needs read / write access to the container for a table space that is owned by the ID that created the database.

### Required Connection

None

### Command Syntax

```
►►──db2untag──-f──filename────────────────────────────────────────────►◄
```

### Command Parameters

**-f filename**
: Specifies the fully qualified name of the table space container from which the DB2 tag is to be removed.

### Usage Notes

An SQLCODE -294 (Container in Use error) is sometimes returned from create database or from create or alter table space operations, usually indicating a specification error on the operating system resource name when the container is already in use by another table space. A container can be used by only one table space at a time.

A system or database administrator who finds that the database which last used the container has been deleted, can use the **db2untag** tool if the container's tag was not removed. If the container is to be released, do one of the following:

- For SMS containers, remove the directory and its contents using the appropriate delete commands.

- For DMS raw containers, either delete the file or device, or let **db2untag** remove the container tag. The tool will leave such a DMS container otherwise unmodified.

## See Also

"CREATE DATABASE" on page 187.

## db2vexp - Dynamic Visual Explain

Explains a dynamic SQL statement, producing an access plan graph. Creates explain tables if they do not exist.

### Authorization

The authorization rules are those defined for the SQL statement specified in the **db2vexp** command. For example, if a DELETE statement was used as the *explainableStatementText*, then the authorization rules for a DELETE statement apply. The current authorization ID must have INSERT privilege on the explain tables. If explain tables do not exist, the current authorization ID must also have CREATETAB privilege on the database. The explain tables will be created automatically.

### Required Connection

None

### Command Syntax

```
►►──db2vexp──-db──databaseName──-sql──explainableStatementText──────────────────────►

   ┌─────────────────────────────┐
   ▼                             │
►────┬─-withsnapshot──────────────┬┴──────────────────────────────────────────────►◄
     ├─-queryno──queryNumber──────┤
     ├─-querytag──queryTag────────┤
     ├─-opt──optimizationClass────┤
     ├─-user──userid──────────────┤
     ├─-password──password────────┤
     └─-h─────────────────────────┘
```

### Command Parameters

**-db databaseName**
: The name of the database where a connection will be made.

**-sql explainableStatementText**
: The statement text to be dynamically explained, enclosed by double quotation marks.

**-withsnapshot**
: Generates more explain details.

**-queryno queryNumber**
: A query number to be associated with the dynamic explain.

**-querytag queryTag**
A query tag to be associated with the dynamic explain. It can be a maximum of twenty characters enclosed by double quotation marks.

**-opt optimizationClass**
Optimization class. Valid values are 0, 1, 2, 3, 5, 7, and 9. See SET CURRENT QUERY OPTIMIZATION in the *SQL Reference*.

**-user userid**
Specifies the user ID used to connect to the database.

**-password password**
Specifies the password used to connect to the database.

**-h** Displays help information. When this option is specified, all other options are ignored and only the help is displayed. On Windows platforms, /? can also be used to specify the help option.

## Usage Notes

A slash (/) can be used instead of a hyphen (-) to prefix a keyword.

# Chapter 2. Command Line Processor (CLP)

This chapter explains how to invoke and use the command line processor, and describes CLP options.

## Command Line Processor Invocation

The **db2** command starts the command line processor. The CLP is used to execute database utilities, SQL statements and online help. It offers a variety of command options, and can be started in:

- Interactive input mode, characterized by the **`db2 =>`** input prompt
- Command mode, where each command must be prefixed by `db2`
- Batch mode, which uses the `-f` file input option.

**Note:** On Windows NT, "db2cmd - Open DB2 Command Window" on page 26 opens the CLP-enabled DB2 window, and initializes the DB2 command line environment. Issuing this command is equivalent to clicking on the *DB2 Command Window* icon.

"QUIT" on page 402 stops the command line processor. "TERMINATE" on page 484 also stops the command line processor, but removes the associated back-end process and frees any memory that is being used. TERMINATE is recommended if the database has been stopped, or if database configuration parameters have been changed.

**Note:** Existing connections should be reset before terminating the CLP.

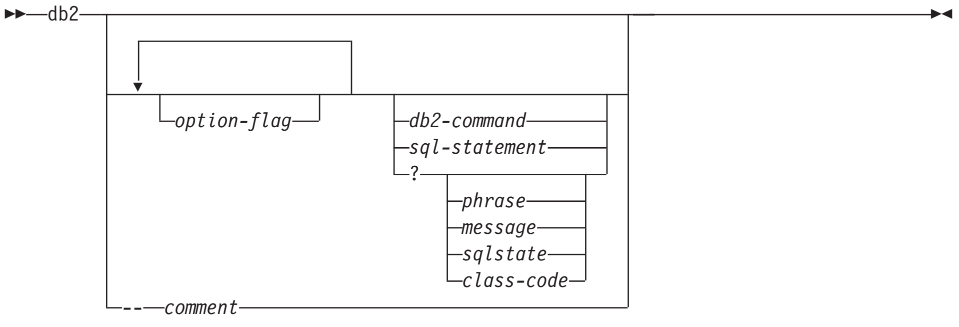
The shell command (**!**), allows operating system commands to be executed from the interactive or the batch mode on UNIX based systems, and on OS/2 or the Windows operating system (`!ls` on UNIX, and `!dir` on OS/2 or the Windows operating system, for example).

**Note:** Shell command support is not available on Windows 3.1.

### Authorization

None

## Command Syntax



## Command Parameters

**option-flag**
: For a summary of valid CLP option flags, see Table 1 on page 97.

**db2-command**
: Specifies a DB2 command. For a description of DB2 commands, see "Chapter 3. CLP Commands" on page 111.

**sql-statement**
: Specifies an SQL statement.

**?** Requests CLP general help.

**? phrase**
: Requests the help text associated with a specified command or topic. If the database manager cannot find the requested information, it displays the general help screen.

  `? options` requests a description and the current settings of the CLP options. `? help` requests information about reading the online help syntax diagrams.

**? message**
: Requests help for a message specified by a valid SQLCODE (`? sql10007n`, for example).

**? sqlstate**
: Requests help for a message specified by a valid SQLSTATE.

**? class-code**
: Requests help for a message specified by a valid class-code.

**-- comment**
: Input that begins with the comment characters `--` is treated as a comment by the command line processor.

**Note:** In each case, a blank space must separate the question mark (?) from the variable name.

## Command Line Processor Options

The CLP command options can be specified by setting the command line processor **DB2OPTIONS** environment variable (which must be in uppercase), or with command line flags.

Users can set options for an entire session using **DB2OPTIONS**.

View the current settings for the option flags and the value of **DB2OPTIONS** using "LIST COMMAND OPTIONS" on page 297. Change an option setting from the interactive input mode or a command file using "UPDATE COMMAND OPTIONS" on page 499.

The command line processor sets options in the following order:

1. Sets up default options.
2. Reads **DB2OPTIONS** to override the defaults.
3. Reads the command line to override **DB2OPTIONS**.
4. Accepts input from UPDATE COMMAND OPTIONS as a final interactive override.

Table 1 summarizes the CLP option flags. These options can be specified in almost any sequence and combination. To turn an option on, prefix the corresponding option letter with a minus sign (-). To turn an option off, either prefix the option letter with a minus sign and follow the option letter with another minus sign, or prefix the option letter with a plus sign (+). For example, `-c` turns the auto-commit option on, and either `-c-` or `+c` turns it off. These option letters are not case sensitive, that is, `-a` and `-A` are equivalent.

Table 1. CLP Command Options

| **Option Flag** | **Description** | **Default Setting** |
|---|---|---|
| -a | This option tells the command line processor to display SQLCA data. | OFF |
| -c | This option tells the command line processor to automatically commit SQL statements. | ON |
| -e{c¦s} | This option tells the command line processor to display SQLCODE or SQLSTATE. These options are mutually exclusive. | OFF |
| -f*filename* | This option tells the command line processor to read command input from a file instead of from standard input. | OFF |
| -l*filename* | This option tells the command line processor to log commands in a history file. | OFF |

Table 1. CLP Command Options (continued)

| Option Flag | Description | Default Setting |
|---|---|---|
| -o | This option tells the command line processor to display output data and messages to standard output. | ON |
| -p | This option tells the command line processor to display a command line processor prompt when in interactive input mode. | ON |
| -r*filename* | This option tells the command line processor to write the report generated by a command to a file. | OFF |
| -s | This option tells the command line processor to stop execution if errors occur while executing commands in a batch file or in interactive mode. | OFF |
| -t | This option tells the command line processor to use a semicolon (;) as the statement termination character. | OFF |
| -tdx | This option tells the command line processor to define and to use x as the statement termination character. | OFF |
| -v | This option tells the command line processor to echo command text to standard output. | OFF |
| -w | This option tells the command line processor to display SQL statement warning messages. | ON |
| -z*filename* | This option tells the command line processor to redirect all output to a file. It is similar to the -r option, but includes any messages or error codes with the output. | OFF |

**Example**

The AIX command:

```
export DB2OPTIONS='+a -c +ec -o -p'
```

sets the following default settings for the session:

```
Display SQLCA   - off
Auto Commit     -  on
Display SQLCODE - off
Display Output  -  on
Display Prompt  -  on
```

The following is a detailed description of these options:

**Show SQLCA Data Option (-a):**
Displays SQLCA data to standard output after executing a DB2 command or an SQL statement. The SQLCA data is displayed instead of an error or success message.

The default setting for this command option is OFF (+a or -a-).

The -o and the -r options affect the -a option; see the option descriptions for details.

**Auto-commit Option (-c):**

This option specifies whether each command or statement is to be treated independently. If set ON (-c), each command or statement is automatically committed or rolled back. If the command or statement is successful, it and all successful commands and statements that were issued before it with autocommit OFF (+c or -c-) are committed. If, however, the command or statement fails, it and all successful commands and statements that were issued before it with autocommit OFF are rolled back. If set OFF (+c or -c-), COMMIT or ROLLBACK must be issued explicitly, or one of these actions will occur when the next command with autocommit ON (-c) is issued.

The default setting for this command option is ON.

The auto-commit option does not affect any other command line processor option.

**Example:** Consider the following scenario:

1. `db2 create database test`
2. `db2 connect to test`
3. `db2 +c "create table a (c1 int)"`
4. `db2 select c2 from a`

The SQL statement in step 4 fails because there is no column named C2 in table A. Since that statement was issued with auto-commit ON (default), it rolls back not only the statement in step 4, but also the one in step 3, because the latter was issued with auto-commit OFF. The command:

```
db2 list tables
```

then returns an empty list.

**Display SQLCODE/SQLSTATE Option (-e):**

The -e{c|s} option tells the command line processor to display the SQLCODE (-ec) or the SQLSTATE (-es) to standard output. Options -ec and -es are not valid in CLP interactive mode.

The default setting for this command option is OFF (+e or -e-).

The -o and the -r options affect the -e option; see the option descriptions for details.

The display SQLCODE/SQLSTATE option does not affect any other command line processor option.

**Example:** To retrieve SQLCODE from the command line processor running on AIX, enter:

```
sqlcode=`db2 -ec +o db2-command`
```

**Read from Input File Option (-f):**

The `-f`*filename* option tells the command line processor to read input from a specified file, instead of from standard input. *Filename* is an absolute or relative file name which may include the directory path to the file. If the directory path is not specified, the current directory is used.

When other options are combined with option `-f`, option `-f` must be specified last. For example:

```
db2 -tvf filename
```

**Note:** This option cannot be changed from within the interactive mode.

The default setting for this command option is OFF (`+f` or `-f-`).

Commands are processed until "QUIT" on page 402 or "TERMINATE" on page 484 is issued, or an end-of-file is encountered.

If both this option and a database command are specified, the command line processor does not process any commands, and an error message is returned.

Input file lines which begin with the comment characters `--` are treated as comments by the command line processor. Comment characters must be the first non-blank characters on a line.

If the `-f`*filename* option is specified, the `-p` option is ignored.

The read from input file option does not affect any other command line processor option.

**Log Commands in History File Option (-l):**

The `-l`*filename* option tells the command line processor to log commands to a specified file. This history file contains records of the commands executed and their completion status. *Filename* is an absolute or relative file name which may include the directory path to the file. If the directory path is not specified, the current directory is used. If the specified file or default file already exists, the new log entry is appended to that file.

When other options are combined with option `-l`, option `-l` must be specified last. For example:

```
db2 -tvl filename
```

The default setting for this command option is OFF (`+l` or `-l-`).

The log commands in history file option does not affect any other command line processor option.

**Display Output Option (-o):**

The `-o` option tells the command line processor to send output data and messages to standard output.

The default setting for this command option is ON.

The interactive mode start-up information is not affected by this option. Output data consists of report output from the execution of the user-specified command, and SQLCA data (if requested).

The following options may be affected by the `+o` option:

- `-r`*filename*: Interactive mode start-up information is not saved.
- `-e`: SQLCODE or SQLSTATE is displayed on standard output even if `+o` is specified.
- `-a`: No effect if `+o` is specified. If `-a`, `+o` and `-r`*filename* are specified, SQLCA information is written to a file.

If both `-o` and `-e` options are specified, the data and either the SQLCODE or the SQLSTATE are displayed on the screen.

If both `-o` and `-v` options are specified, the data is displayed, and the text of each command issued is echoed to the screen.

The display output option does not affect any other command line processor option.

**Display DB2 Interactive Prompt Option (-p):**

The `-p` option tells the command line processor to display the command line processor prompt when the user is in interactive mode.

The default setting for this command option is ON.

Turning the prompt off is useful when commands are being piped to the command line processor. For example, a file containing CLP commands could be executed by issuing:

```
db2 +p < myfile.clp
```

The `-p` option is ignored if the `-f`*filename* option is specified.

The display DB2 interactive prompt option does not affect any other command line processor option.

**Save to Report File Option (-r):**

The `-r`*filename* option causes any output data generated by a command to be written to a specified file, and is useful for capturing a report that would otherwise scroll off the screen. Messages or error codes are not written to the file. *Filename* is an absolute or relative file name which may include the directory path to the file. If the directory path is not specified, the current directory is used. New report entries are appended to the file.

The default setting for this command option is OFF (`+r` or `-r-`).

If the `-a` option is specified, SQLCA data is written to the file.

The `-r` option does not affect the `-e` option. If the `-e` option is specified, SQLCODE or SQLSTATE is written to standard output, not to a file.

If `-r`*filename* is set in **DB2OPTIONS**, the user can set the `+r` (or `-r-`) option from the command line to prevent output data for a particular command invocation from being written to the file.

The save to report file option does not affect any other command line processor option.

**Stop Execution on Command Error Option (-s):**

When commands are issued in interactive mode, or from an input file, and syntax or command errors occur, the `-s` option causes the command line processor to stop execution and to write error messages to standard output.

The default setting for this command option is OFF (`+s` or `-s-`). This setting causes the command line processor to display error messages, continue execution of the remaining commands, and to stop execution only if a system error occurs (return code 8).

The following table summarizes this behavior:

Table 2. CLP Return Codes and Command Execution

| Return Code | `-s` Option Set | `+s` Option Set |
|---|---|---|
| 0 (success) | execution continues | execution continues |
| 1 (0 rows selected) | execution continues | execution continues |
| 2 (warning) | execution continues | execution continues |
| 4 (DB2 or SQL error) | execution stops | execution continues |
| 8 (System error) | execution stops | execution stops |

**Statement Termination Character Option (-t):**

The `-t` option tells the command line processor to use a semicolon (;) as the statement termination character, and disables the backslash (\) line continuation character.

**Note:** This option cannot be changed from within the interactive mode.

The default setting for this command option is OFF (`+t` or `-t-`).

To define a termination character, use -td followed by the chosen termination character. For example, `-tdx` sets `x` as the statement termination character.

The termination character cannot be used to concatenate multiple statements from the command line, since only the last non-blank character on each input line is checked for a termination symbol.

The statement termination character option does not affect any other command line processor option.

**Verbose Output Option (-v):**

The `-v` option causes the command line processor to echo (to standard output) the command text entered by the user prior to displaying the output, and any messages from that command. "ECHO" on page 205 is exempt from this option.

The default setting for this command option is OFF (`+v` or `-v-`).

The `-v` option has no effect if `+o` (or `-o-`) is specified.

The verbose output option does not affect any other command line processor option.

**Show Warning Messages Option (-w):**

The `-w` option tells the command line processor to show SQL statement warning messages.

The default setting for this command option is ON.

**Save all Output to File Option (-z):**

The `-z`*filename* option causes all output generated by a command to be written to a specified file, and is useful for capturing a report that would otherwise scroll off the screen. It is similar to the `-r` option; in this case, however, messages, error codes, and other informational output are also written to the file. *Filename* is an absolute or relative file name which may include the directory path to the file. If the directory path is not specified, the current directory is used. New report entries are appended to the file.

The default setting for this command option is OFF (`+z` or `-z-`).

If the `-a` option is specified, SQLCA data is written to the file.

The `-z` option does not affect the `-e` option. If the `-e` option is specified, SQLCODE or SQLSTATE is written to standard output, not to a file.

If `-z`*filename* is set in **DB2OPTIONS**, the user can set the `+z` (or `-z-`) option from the command line to prevent output data for a particular command invocation from being written to the file.

The save all output to file option does not affect any other command line processor option.

## Command Line Processor Return Codes

When the command line processor finishes processing a command or an SQL statement, it returns an exit (or return) code. These codes are transparent to users executing CLP functions from the command line, but they can be retrieved when those functions are executed from a shell script.

For example, the following Bourne shell script executes the GET DATABASE MANAGER CONFIGURATION command, then inspects the CLP return code:

```
db2 get database manager configuration
if ["$?" = "0"]
then echo "OK!"
fi
```

The return code can be one of the following:

| **Code** | **Description** |
|---|---|
| **0** | DB2 command or SQL statement executed successfully |
| **1** | SELECT or FETCH statement returned no rows |
| **2** | DB2 command or SQL statement warning |
| **4** | DB2 command or SQL statement error |
| **8** | Command line processor system error |

The command line processor does not provide a return code while a user is executing statements from interactive mode, or while input is being read from a file (using the `-f` option).

A return code is available only after the user quits interactive mode, or when processing of an input file ends. In these cases, the return code is the logical OR of the distinct codes returned from the individual commands or statements executed to that point.

For example, if a user in interactive mode issues commands resulting in return codes of 0, 1, and 2, a return code of 3 will be returned after the user quits interactive mode. The individual codes 0, 1, and 2 are not returned. Return code 3 tells the user that during interactive mode processing, one or more commands returned a 1, and one or more commands returned a 2.

A return code of 4 results from a negative SQLCODE returned by a DB2 command or an SQL statement. A return code of 8 results only if the command line processor encounters a system error.

If commands are issued from an input file or in interactive mode, and the command line processor experiences a system error (return code 8), command execution is halted immediately. If one or more DB2 commands or SQL statements end in error (return code 4), command execution stops if the `-s` (Stop Execution on Command Error) option is set; otherwise, execution continues.

## Using the Command Line Processor

The command line processor operates as follows:

- The CLP command (in either case) is typed at the command prompt.
- The command is sent to the command shell by pressing the ENTER key.
- Output is automatically directed to the standard output device.
- Piping and redirection are supported.
- The user is notified of successful and unsuccessful completion.
- Following execution of the command, control returns to the operating system command prompt, and the user may enter more commands.

Before accessing a database, the user must perform preliminary tasks, such as starting DB2 with "START DATABASE MANAGER" on page 475. The user must also connect to a database before it can be queried. Connect to a database by doing one of the following:

- Issue the SQL CONNECT TO *database* statement
- Establish an implicit connection to the default database defined by the environment variable **DB2DBDFT**.

For information about working with tables within a database, see the *SQL Reference*.

If a command exceeds the character limit allowed at the command prompt, a backslash (\) can be used as the line continuation character. When the command line processor encounters the line continuation character, it reads the next line and concatenates the characters contained on both lines. Alternatively, the `-t` option can be used to set a line termination character (see page 102). In this case, the line continuation character is invalid, and all statements and commands must end with the line termination character.

The command line processor recognizes a string called `NULL` as a null string. Fields that have been set previously to some value can later be set to NULL. For example,

```
db2 update database manager configuration using tm_database NULL
```

sets the *tm_database* field to NULL. This operation is case sensitive. A lowercase `null` is not interpreted as a null string, but rather as a string containing the letters `null`.

## Using the Command Line Processor in Command Files

CLP requests to the database manager can be imbedded in a shell script command file. The following example shows how to enter the CREATE TABLE statement in a shell script command file:

```
db2 "create table mytable (name VARCHAR(20), color CHAR(10))"
```

For more information about commands and command files, see the appropriate operating system manual.

## Command Line Processor Design

The command line processor consists of two processes: the front-end process (the DB2 command), which acts as the user interface, and the back-end process (db2bp), which maintains a database connection.

### Maintaining Database Connections

Each time that **db2** is invoked, a new front-end process is started. The back-end process is started by the first **db2** invocation, and can be explicitly terminated with “TERMINATE” on page 484. All front-end processes with the same parent are serviced by a single back-end process, and therefore share a single database connection.

For example, the following **db2** calls from the same operating system command prompt result in separate front-end processes sharing a single back-end process, which holds a database connection throughout:

- db2 'connect to sample’,
- db2 'select * from org’,
- . foo (where foo is a shell script containing DB2 commands), and
- db2 -tf myfile.clp.

The following invocations from the same operating system prompt result in separate database connections because each has a distinct parent process, and therefore a distinct back-end process:

- foo
- . foo &
- foo &
- sh foo

### Communication between Front-end and Back-end Processes

The front-end process and back-end processes communicate through three message queues: a request queue, an input queue, and an output queue.

### Environment Variables

The following environment variables offer a means of configuring communication between the two processes:

Table 3. Environment Variables

| Variable | Minimum | Maximum | Default |
|---|---|---|---|
| DB2BQTIME | 1 second | 5294967295 | 1 second |
| DB2BQTRY | 0 tries | 5294967295 | 60 tries |
| DB2RQTIME | 1 second | 5294967295 | 5 seconds |
| DB2IQTIME | 1 second | 5294967295 | 5 seconds |

**DB2BQTIME**
When the command line processor is invoked, the front-end process checks if the back-end process is already active. If it is active, the front-end process re-establishes a connection to it. If it is not active, the front-end process activates it. The front-end process then idles for the duration specified by the **DB2BQTIME** variable, and checks again. The front-end process continues to check for the number of times specified by the **DB2BQTRY** variable, after which, if the back-end process is still not active, it times out and returns an error message.

**DB2BQTRY**
works in conjunction with the **DB2BQTIME** variable, and specifies the number of times the front-end process tries to determine whether the back-end process is active.

The values of **DB2BQTIME** and **DB2BQTRY** can be increased during peak periods to optimize query time.

**DB2RQTIME**
Once the back-end process has been started, it waits on its request

queue for a request from the front-end. It also waits on the request queue between requests initiated from the command prompt.

The **DB2RQTIME** variable specifies the length of time the back-end process waits for a request from the front-end process. At the end of this time, if no request is present on the request queue, the back-end process checks whether the parent of the front-end process still exists, and terminates itself if it does not exist. Otherwise, it continues to wait on the request queue.

**DB2IQTIME**
When the back-end process receives a request from the front-end process, it sends an acknowledgement to the front-end process indicating that it is ready to receive input via the input queue. The back-end process then waits on its input queue. It also waits on the input queue while a batch file (specified with the `-f` option) is executing, and while the user is in interactive mode.

The **DB2IQTIME** variable specifies the length of time the back-end process waits on the input queue for the front-end process to pass the commands. After this time has elapsed, the back-end process checks whether the front-end process is active, and returns to wait on the request queue if the front-end process no longer exists. Otherwise, the back-end process continues to wait for input from the front-end process.

To view the values of these environment variables, use “LIST COMMAND OPTIONS” on page 297.

The back-end environment variables inherit the values set by the front-end process at the time the back-end process is initiated. However, if the front-end environment variables are changed, the back-end process will not inherit these changes. The back-end process must first be terminated, and then restarted (by issuing the **db2** command) to inherit the changed values.

An example of when the back-end process must be terminated is provided by the following scenario:

1. User A logs on, issues some CLP commands, and then logs off without issuing TERMINATE.
2. User B logs on using the same window.
3. When user B issues certain CLP commands, they fail with message DB21016 (system error).

The back-end process started by user A is still active when user B starts using the CLP, because the parent of user B’s front-end process (the operating system window from which the commands are issued) is still active. The back-end process attempts to service the new commands issued by user B;

however, user B’s front-end process does not have enough authority to use the message queues of the back-end process, because it needs the authority of user A, who created that back-end process. A CLP session must end with a TERMINATE command before a user starts a new CLP session using the same operating system window. This creates a fresh back-end process for each new user, preventing authority problems, and setting the correct values of environment variables (such as **DB2INSTANCE**) in the new user’s back-end process.

## CLP Usage Notes

Commands can be entered either in uppercase or in lowercase from the command prompt. However, parameters that are case sensitive to DB2 must be entered in the exact case desired. For example, the *comment-string* in the WITH clause of the CHANGE DATABASE COMMENT command is a case sensitive parameter.

Delimited identifiers are allowed in SQL statements. For more detailed information on the use of delimited identifiers in SQL statements, see the *SQL Reference*.

Special characters, or metacharacters (such as $ & * ( ) ; < > ? \ ' ") are allowed within CLP commands. If they are used outside the CLP interactive mode, or the CLP batch input mode, these characters are interpreted by the operating system shell. Quotation marks or an escape character are required if the shell is not to take any special action.

For example, when executed inside an AIX Korn shell environment,

```
db2 select * from org where division > 'Eastern'
```

is interpreted as ″select <the names of all files> from org where division″. The result, an SQL syntax error, is redirected to the file `Eastern`. The following syntax produces the correct output:

```
db2 "select * from org where division > 'Eastern'"
```

Special characters vary from platform to platform. In the AIX Korn shell, the above example could be rewritten using an escape character (\), such as *, \>, or \'. In the OS/2 shell, * or \' results in a syntax error.

Most operating system environments allow input and output to be redirected. For example, if a connection to the SAMPLE database has been made, the following request queries the STAFF table, and sends the output to a file named `staflist.txt` in the `mydata` directory:

```
db2 "select * from staff" > mydata/staflist.txt
```

## CLP Usage Notes

For environments such as Windows 3.1, where output redirection is not supported, CLP options can be used. For example, the request can be rewritten as

```
db2 -r mydata\staflist.txt "select * from staff"

db2 -z mydata\staflist.txt "select * from staff"
```

For more information about CLP options for Windows, see the *Installation and Configuration Supplement* book.

The command line processor is not a programming language. For example, it does not support host variables, and the statement,

```
db2 connect to :HostVar in share mode
```

is syntactically incorrect, because `:HostVar` is not a valid database name.

The command line processor represents SQL NULL values as hyphens (-). If the column is numeric, the hyphen is placed at the right of the column. If the column is not numeric, the hyphen is at the left. For information about using the command line processor with DB2 Connect and host databases, see the *DB2 Connect User's Guide*.

# Chapter 3. CLP Commands

This chapter describes the DB2 commands in alphabetical order. These commands are used to control the system interactively.

**Note:** Slashes (/) in directory paths are specific to UNIX based systems, and are equivalent to back slashes (\) in directory paths on OS/2 and Windows operating systems.

For information about how the command descriptions are organized, see "How the Command Descriptions are Organized" on page 2.

## DB2 CLP Commands

The following table lists the CLP commands grouped by functional category:

Table 4. DB2 CLP Commands

| **CLP Session Control** |
|---|
| "LIST COMMAND OPTIONS" on page 297 |
| "UPDATE COMMAND OPTIONS" on page 499 |
| "CHANGE ISOLATION LEVEL" on page 185 |
| "SET RUNTIME DEGREE" on page 469 |
| "TERMINATE" on page 484 |
| "QUIT" on page 402 |
| **Database Manager Control** |
| "START DATABASE MANAGER" on page 475 |
| "STOP DATABASE MANAGER" on page 480 |
| "GET DATABASE MANAGER CONFIGURATION" on page 236 |
| "RESET DATABASE MANAGER CONFIGURATION" on page 435 |
| "UPDATE DATABASE MANAGER CONFIGURATION" on page 503 |
| "GET ADMIN CONFIGURATION" on page 217 |
| "RESET ADMIN CONFIGURATION" on page 430 |
| "UPDATE ADMIN CONFIGURATION" on page 494 |
| **Database Control** |
| "RESTART DATABASE" on page 439 |

Table 4. DB2 CLP Commands (continued)

| |
|---|
| "CREATE DATABASE" on page 187 |
| "DROP DATABASE" on page 202 |
| "MIGRATE DATABASE" on page 370 |
| "LIST INDOUBT TRANSACTIONS" on page 314 |
| "ACTIVATE DATABASE" on page 116 |
| "DEACTIVATE DATABASE" on page 194 |
| "LIST DRDA INDOUBT TRANSACTIONS" on page 308 |
| **Database Directory Management** |
| "CATALOG DATABASE" on page 153 |
| "UNCATALOG DATABASE" on page 485 |
| "CATALOG DCS DATABASE" on page 157 |
| "UNCATALOG DCS DATABASE" on page 487 |
| "CHANGE DATABASE COMMENT" on page 183 |
| "LIST DATABASE DIRECTORY" on page 299 |
| "LIST DCS DIRECTORY" on page 306 |
| "CATALOG GLOBAL DATABASE" on page 160 |
| **ODBC Management** |
| "CATALOG ODBC DATA SOURCE" on page 178 |
| "LIST ODBC DATA SOURCES" on page 325 |
| "UNCATALOG ODBC DATA SOURCE" on page 493 |
| "GET CLI CONFIGURATION" on page 222 |
| "UPDATE CLI CONFIGURATION" on page 497 |
| **Client/Server Directory Management** |
| "CATALOG LOCAL NODE" on page 171 |
| "CATALOG NAMED PIPE NODE" on page 173 |
| "CATALOG APPC NODE" on page 145 |
| "CATALOG APPCLU NODE" on page 148 |
| "CATALOG APPN NODE" on page 150 |
| "CATALOG IPX/SPX NODE" on page 162 |
| "CATALOG NETBIOS NODE" on page 175 |
| "CATALOG TCP/IP NODE" on page 179 |
| "UNCATALOG NODE" on page 491 |
| "LIST NODE DIRECTORY" on page 319 |

Table 4. DB2 CLP Commands (continued)

| **Network Support** |
|---|
| "REGISTER" on page 414 |
| "DEREGISTER" on page 196 |
| "UPDATE LDAP NODE" on page 505 |
| "CATALOG LDAP DATABASE" on page 165 |
| "UNCATALOG LDAP DATABASE" on page 488 |
| "CATALOG LDAP NODE" on page 169 |
| "UNCATALOG LDAP NODE" on page 490 |
| "REFRESH LDAP" on page 413 |
| **Database Configuration** |
| "GET DATABASE CONFIGURATION" on page 225 |
| "RESET DATABASE CONFIGURATION" on page 433 |
| "UPDATE DATABASE CONFIGURATION" on page 501 |
| **Recovery** |
| "BACKUP DATABASE" on page 124 |
| "RECONCILE" on page 406 |
| "RESTORE DATABASE" on page 441 |
| "ROLLFORWARD DATABASE" on page 451 |
| "LIST HISTORY" on page 311 |
| "PRUNE HISTORY/LOGFILE" on page 395 |
| "UPDATE RECOVERY HISTORY FILE" on page 510 |
| **Operational Utilities** |
| "FORCE APPLICATION" on page 215 |
| "LIST PACKAGES/TABLES" on page 327 |
| "REORGCHK" on page 422 |
| "REORGANIZE TABLE" on page 419 |
| "RUNSTATS" on page 461 |
| **Database Monitoring** |
| "GET MONITOR SWITCHES" on page 252 |
| "UPDATE MONITOR SWITCHES" on page 508 |
| "GET DATABASE MANAGER MONITOR SWITCHES" on page 249 |
| "GET SNAPSHOT" on page 254 |
| "RESET MONITOR" on page 437 |

Table 4. DB2 CLP Commands (continued)

| |
|---|
| “LIST ACTIVE DATABASES” on page 294 |
| “LIST APPLICATIONS” on page 295 |
| “LIST DCS APPLICATIONS” on page 303 |
| **Data Utilities** |
| “EXPORT” on page 206 |
| “IMPORT” on page 273 |
| “LOAD” on page 338 |
| “LOAD QUERY” on page 368 |
| **Application Preparation** |
| “PRECOMPILE PROGRAM” on page 372 |
| “BIND” on page 131 |
| “REBIND” on page 403 |
| **Remote Server Utilities** |
| “ATTACH” on page 121 |
| “DETACH” on page 201 |
| **Table Space Management** |
| “LIST TABLESPACE CONTAINERS” on page 330 |
| “SET TABLESPACE CONTAINERS” on page 471 |
| “LIST TABLESPACES” on page 332 |
| “QUIESCE TABLESPACES FOR TABLE” on page 399 |
| **Node Management** |
| “ADD NODE” on page 119 |
| “DROP NODE VERIFY” on page 204 |
| “LIST NODES” on page 324 |
| **Nodegroup Management** |
| “LIST NODEGROUPS” on page 322 |
| “REDISTRIBUTE NODEGROUP” on page 409 |
| **Additional Commands** |
| “ADD DATALINKS MANAGER” on page 118 |
| “DESCRIBE” on page 198 |
| “ECHO” on page 205 |
| “GET AUTHORIZATIONS” on page 220 |
| “GET CONNECTION STATE” on page 224 |

Table 4. DB2 CLP Commands (continued)

| |
|---|
| “GET INSTANCE” on page 251 |
| “HELP” on page 271 |
| “INVOKE STORED PROCEDURE” on page 292 |
| “LIST DATALINKS MANAGERS” on page 310 |
| “QUERY CLIENT” on page 397 |
| “SET CLIENT” on page 465 |
| “INITIALIZE TAPE” on page 291 |
| “REWIND TAPE” on page 450 |
| “SET TAPE POSITION” on page 474 |

## ACTIVATE DATABASE

Activates the specified database and starts up all necessary database services, so that the database is available for connection and use by any application.

### Scope

This command activates the specified database on all nodes within the system. If one or more of these nodes encounters an error during activation of the database, a warning is returned. The database remains activated on all nodes on which the command has succeeded.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

None

### Command Syntax

```
►►──ACTIVATE──┬─DATABASE─┬──database-alias──────────────────────────────►
              └─DB───────┘

►──┬──────────────────────────────────────┬──────────────────────────►◄
   └─USER──username──┬──────────────────┬─┘
                     └─USING──password──┘
```

### Command Parameters

**database-alias**
: Specifies the alias of the database to be started.

**USER username**
: Specifies the user starting the database.

**USING password**
: Specifies the password for the user name.

### Usage Notes

If a database has not been started, and a CONNECT TO (or an implicit connect) is issued in an application, the application must wait while the database manager starts the required database, before it can do any work with that database. However, once the database is started, other applications can simply connect and use it without spending time on its start up.

Database administrators can use ACTIVATE DATABASE to start up selected databases. This eliminates any application time spent on database initialization.

Databases initialized by ACTIVATE DATABASE can be shut down by "DEACTIVATE DATABASE" on page 194, or by "STOP DATABASE MANAGER" on page 480.

If a database was started by a CONNECT TO (or an implicit connect) and subsequently an ACTIVATE DATABASE is issued for that same database, then DEACTIVATE DATABASE must be used to shut down that database. If ACTIVATE DATABASE was not used to start the database, the database will shut down when the last application disconnects.

ACTIVATE DATABASE behaves in a similar manner to a CONNECT TO (or an implicit connect) when working with a database requiring a restart (for example, database in an inconsistent state). The database will be restarted before it can be initialized by ACTIVATE DATABASE. Restart will only be performed if the database is configured to have AUTORESTART ON.

**Note:** The application issuing the ACTIVATE DATABASE command cannot have an active database connection to any database.

## See Also

"DEACTIVATE DATABASE" on page 194

"STOP DATABASE MANAGER" on page 480.

## ADD DATALINKS MANAGER

Adds a DB2 Data Links Manager (host name) to the list of registered DB2 Data Links Managers for a specified database.

### Authorization

None

### Command Syntax

```
►►──ADD DATALINKS MANAGER FOR──┬─DATABASE─┬──dbname──USING NODE──hostname──────────────►
                               └─DB───────┘

►─PORT──port-number──────────────────────────────────────────────────────────────────►◄
```

### Command Parameters

**DATABASE dbname**
Specifies a database name.

**USING NODE hostname**
Specifies a fully qualified host name, or the IP address (but not both), of the DB2 Data Links Manager server.

**PORT port-number**
Specifies the port number that has been reserved for communications from the DB2 server to the DB2 Data Links Manager server.

### See Also

"LIST DATALINKS MANAGERS" on page 310.

## ADD NODE

Adds a new node to the parallel database system. This command creates database partitions for all databases currently defined in the MPP server on the new node. The user can specify the source node for any temporary table spaces to be created with the databases, or specify that no temporary table spaces are to be created. The command must be issued from the node that is being added, and can only be issued on an MPP server.

### Scope

This command only affects the node on which it is executed.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──ADD NODE──┬──────────────────────────┬─────────────────────────────────►◄
              ├─LIKE NODE──node-number───┤
              └─WITHOUT TABLESPACES──────┘
```

### Command Parameters

**LIKE NODE node-number**
: Specifies that the containers for the temporary table spaces will be the same as the containers on the specified *node-number* for each database in the instance. The node specified must be a node that is already in the `db2nodes.cfg` file.

**WITHOUT TABLESPACES**
: Specifies that containers for the temporary table spaces are not created for any of the databases. The ALTER TABLESPACE statement must be used to add temporary table space containers to each database before the database can be used.

**Note:** If no option is specified, containers for the temporary table spaces will be the same as the containers on the catalog node for each database. The catalog node may be a different node for each database in the MPP system.

## Usage Notes

Before adding a new node, ensure that there is sufficient storage for the containers that must be created for all existing databases on the system.

The add node operation creates an empty database partition on the new node for every database that exists in the instance. The configuration parameters for the new database partitions are set to the default value.

If an add node operation fails while creating a database partition locally, it enters a clean-up phase, in which it locally drops all databases that have been created. This means that the database partitions are removed only from the node being added (that is, the local node). Existing database partitions remain unaffected on all other nodes. If this fails, no further clean up is done, and an error is returned.

The database partitions on the new node cannot be used to contain user data until after the ALTER NODEGROUP statement has been used to add the node to a nodegroup. For details, see the *SQL Reference*.

This command will fail if a create database or a drop database operation is in progress. The command can be reissued once the operation has completed.

If temporary table spaces are to be created with the database partitions, ADD NODE may have to communicate with another node in the MPP system in order to retrieve the table space definitions. The *start_stop_time* database manager configuration parameter is used to specify the time, in minutes, by which the other node must respond with the table space definitions. If this time is exceeded, the command fails. Increase the value of *start_stop_time*, and reissue the command.

## See Also

"START DATABASE MANAGER" on page 475.

## ATTACH

Enables an application to specify the instance at which instance-level commands (CREATE DATABASE and FORCE APPLICATION, for example) are to be executed. This instance may be the current instance, another instance on the same workstation, or an instance on a remote workstation.

### Authorization

None

### Required Connection

None. This command establishes an instance attachment.

### Command Syntax

```
►►──ATTACH──┬───────────────┬──────────────────────────────────────────────►
            └─TO──nodename──┘

►─┬───────────────────────────────────────────────────────────────────┬───►◄
  └─USER──username──┬───────────────────────────────────────────────┬─┘
                    ├─USING──password──┬──────────────────────────────────┬─┤
                    │                  └─NEW──password──CONFIRM──password─┘ │
                    └─CHANGE PASSWORD──────────────────────────────────────┘
```

### Command Parameters

**TO nodename**
: Alias of the instance to which the user wants to attach. This instance must have a matching entry in the local node directory. The only exception to this is the local instance (as specified by the **DB2INSTANCE** environment variable) which may be specified as the object of an attach, but which cannot be used as a node name in the node directory.

**USER username**
: Specifies the authentication identifier.

**USING password**
: Specifies the password for the user name. If a user name is specified, but a password is *not* specified, the user is prompted for the current password. The password is not displayed at entry.

**NEW password**
: Specifies the new password that is to be assigned to the user name.

Passwords can be up to 18 characters in length. The system on which the password will be changed depends on how user authentication has been set up.

**CONFIRM password**
A string that must be identical to the new password. This parameter is used to catch entry errors.

**CHANGE PASSWORD**
If this option is specified, the user is prompted for the current password, a new password, and for confirmation of the new password. Passwords are not displayed at entry.

## Examples

Catalog two remote nodes:

```
db2 catalog tcpip node node1 remote freedom server server1
db2 catalog tcpip node node2 remote flash server server1
```

Attach to the first node, force all users, and then detach:

```
db2 attach to node1
db2 force application all
db2 detach
```

Attach to the second node, and see who is on:

```
db2 attach to node2
db2 list applications
```

After the command returns agent IDs 1, 2 and 3, force 1 and 3, and then detach:

```
db2 force application (1, 3)
db2 detach
```

Attach to the current instance (not necessary, will be implicit), force all users, then detach (AIX only):

```
db2 attach to $DB2INSTANCE
db2 force application all
db2 detach
```

## Usage Notes

If *nodename* is omitted from the command, information about the current state of attachment is returned.

If ATTACH has not been executed, instance-level commands are executed against the current instance, specified by the **DB2INSTANCE** environment variable.

## See Also

“DETACH” on page 201.

## BACKUP DATABASE

Creates a backup copy of a database or a table space.

### Scope

This command only affects the node on which it is executed.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Database. This command automatically establishes a connection to the specified database.

### Command Syntax

```
►►──BACKUP──┬─DATABASE─┬──database-alias─────────────────────────────────────►
            └─DB───────┘

►─┬──────────────────────────────────┬────────────────────────────────────────►
  └─USER──username─┬────────────────┬┘
                   └─USING──password─┘

►─┬────────────────────────────────────────────┬─┬────────┬──────────────────►
  │                 ┌─,───────────────┐        │ └─ONLINE─┘
  │                 ▼                 │        │
  └─TABLESPACE──(──────tablespace-name┴────)───┘

►─┬─────────────────────────────────────────────────────────┬────────────────►
  ├─USE ADSM─┬──────────────────────────────────┬───────────┤
  │          └─OPEN──num-sessions──SESSIONS─────┘           │
  │      ┌─,──────┐                                         │
  │      ▼        │                                         │
  ├─TO─────┬─dir─┬┴─────────────────────────────────────────┤
  │        └─dev─┘                                          │
  └─LOAD──library-name─┬───────────────────────────────────┬┘
                       └─OPEN──num-sessions──SESSIONS──────┘
```

```
►─┬──────────────────────────────┬──┬────────────────────────┬──┬───────────────────┬──────────►
  └─WITH──num-buffers──BUFFERS───┘  └─BUFFER──buffer-size────┘  └─PARALLELISM──n────┘

►─┬──────────────────────┬──────────────────────────────────────────────────────────────────►◄
  └─WITHOUT PROMPTING────┘
```

## Command Parameters

**DATABASE database-alias**
Specifies the alias of the database to back up.

**USER username**
Identifies the user name under which to back up the database.

**USING password**
The password used to authenticate the user name. If the password is omitted, the user is prompted to enter it.

**TABLESPACE tablespace-name**
A list of names used to specify the table spaces to be backed up.

**ONLINE**
Specifies online backup. The default is offline backup. Online backups are only available for databases configured with *logretain* or *userexit* enabled.

**Note:** An online backup operation may time out if there is an IX lock on `sysibm.systables`, because the DB2 backup utility requires an S lock on objects containing LOBs.

**USE ADSM**
Specifies that the backup is to use ADSM managed output.

**OPEN num-sessions SESSIONS**
The number of I/O sessions to be used with ADSM or another product.

**TO dir/dev**
A list of directory or tape device names. The full path on which the directory resides must be specified. The target must reside on the database server. This parameter may be repeated to specify the target directories and devices that the backup image will span. If more than one target is specified (target1, target2, and target3, for example), target1 will be opened first. The media header and special files (including the configuration file, table space table, and history file) are placed in target1. All remaining targets are opened, and are then used in parallel during the backup.

Use of tape devices or floppy disks may generate messages and prompts for user input. Valid response options are:

**c** Continue. Continue using the device that generated the warning message (for example, when a new tape has been mounted)

**d** Device terminate. Stop using *only* the device that generated the warning message (for example, when there are no more tapes)

**t** Terminate. Abort the backup or restore utility.

Tape is not supported on OS/2. On OS/2, `0` or `0:` can be specified to cause the backup operation to call the user exit program (see the *Administration Guide*). This option is not valid on any other platform.

**LOAD library-name**
The name of the shared library (DLL on OS/2 or the Windows operating system) containing the vendor backup and restore I/O functions to be used. It can contain the full path. If the full path is not given, it will default to the path on which the user exit program resides.

**WITH num-buffers BUFFERS**
The number of buffers to be used.

**BUFFER buffer-size**
The size, in pages, of the buffer used when building the backup image. The minimum value for this parameter is 8 pages; the default value is 1024 pages. If a buffer size of zero is specified, the value of the database manager configuration parameter *backbufsz* must be set to 16.

To use tape devices, DB2 users on SCO UnixWare 7 must specify a buffer size of 16.

**PARALLELISM n**
Specifies the number of buffer manipulators to be spawned during the backup process. The default value is 1.

**WITHOUT PROMPTING**
Specifies that the backup will run unattended, and that any actions which normally require user intervention will return an error message.

## Examples

```
db2 backup database sample use adsm open 2 sessions with 4 buffers

db2 backup database payroll tablespace syscatspace, userspace1 to
   /dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```

## Usage Notes

### Database Level Backup

**Note:** Backup each database on a regular basis. If a database becomes damaged or corrupted, it can be returned to the state of the backed-up copy (see “RESTORE DATABASE” on page 441).

If a successfully restored database was enabled for roll-forward recovery at the time of the backup operation, it can be returned to the state it was in prior to the occurrence of damage (see “ROLLFORWARD DATABASE” on page 451).

The backup can be directed to fixed disk, diskette, tape, ADSM utility, or to other vendor products enabled for DB2.

On UNIX based systems, a backup file name consists of the concatenation of several units of information separated by periods:

**dbname.type.instance.nodexxxx.catnxxxx.yyyymmddhhmmss.seq**

| | |
|---|---|
| **dbname** | 1 to 8 character database alias. |
| **type** | Type of backup taken (`0` for full database, or `3` for table space level backup). |
| **instance** | 1 to 8 character database instance name. |
| **nodexxxx** | The number of the node. In non-partitioned database systems, this is always zero (`NODE0000`). In a partitioned database system, it is `NODE`*xxxx*, where *xxxx* is the number assigned to the node in the `db2nodes.cfg` file. |
| **catnxxxx** | The number of the catalog node for the database. In non-partitioned database systems, this is always zero (`CATN0000`). In a partitioned database system, it is `CATN`*xxxx*, where *xxxx* is the number assigned to the node in the `db2nodes.cfg` file. |
| **yyyymmdd** | Date (year month day). |
| **hhmmss** | Time (hour minute second). |
| **seq** | A file extension consisting of a 3-digit sequence number. |

In addition to fixed disk, tape, ADSM and other vendors, the backup may be directed to diskettes. Since there is no general tape support on OS/2 or the Windows operating system, each type of tape device requires a unique device driver.

To back up to the FAT file system on OS/2 or the Windows operating system, users must conform to the 8.3 naming restriction. The file is placed in a 5-level subdirectory tree as follows:

**dbname.type\db2instance.nodexxx\catnxxx\yyyymmdd\hhmmss.seq**

| | |
|---|---|
| **dbname** | 1 to 8 character database alias. |
| **type** | Type of backup taken. `0` for full database, `3` for table space level backup, `4` for copy from a table load. |
| **db2instance** | 1 to 8 character database instance name. |
| **nodexxx** | The number of the node. In non-partitioned database systems, this is always zero (`NODE000`). In a partitioned database system, it is `NODE`*xxx*, where *xxx* is the number assigned to the node in the `db2nodes.cfg` file. |
| **catnxxx** | The number of the catalog node for the database. In non-partitioned database systems, this is always zero (`CATN000`). In a partitioned database system, it is `CATN`*xxx*, where *xxx* is the number assigned to the node in the `db2nodes.cfg` file. |
| **yyyymmdd** | Date (year month day). |
| **hhmmss** | Time (hour minute second). |
| **seq** | A file extension consisting of a 3-digit sequence number. |

Online backups are permitted only if roll-forward recovery is enabled. The utility can perform an online backup while it is being accessed and modified by other applications.

To perform an offline backup, the utility must be able to connect to the database in exclusive mode. BACKUP fails if any application, including the calling application, is already connected to the database. If the connection is successful, BACKUP locks out other applications until the backup is completed.

Execute BACKUP DATABASE offline when the database is not currently needed.

An offline backup operation will fail if the database is not in a consistent state. If the database is inconsistent, it must be restarted to be brought back to a consistent state through crash recovery before BACKUP is executed (see "RESTART DATABASE" on page 439, and a description of the *autorestart* configuration parameter in the *Administration Guide*). If the database is in a partially restored state after a system failure during restoration, the restore operation must be successfully rerun before a backup can be executed. If the database has been restored and a roll-forward operation is needed, the database must be rolled forward to a consistent state before it can be backed up.

If a database is changed from roll-forward disabled to roll-forward enabled state, either the *logretain* or the *userexit* database configuration options must be enabled before a backup of the database can be made (see “GET DATABASE CONFIGURATION” on page 225).

**Table Space Level Backup**

A table space level backup contains one or more table spaces for a database, specified when the command is executed. A table space level backup can be taken online or offline, but the database must be configured with roll-forward recovery enabled.

Table space level backup can be used to recover from problems that only affect specific table spaces. While this recovery is taking place, all other table spaces are available for processing.

To ensure that restored table spaces are synchronized with the rest of the database, the table spaces must be rolled forward to the end of the log (or to the point where the table spaces were last used). For this reason, table space level backup and restore can only be performed if roll-forward recovery is enabled. If roll-forward recovery is disabled at any time after a table space level backup is executed, it will not be possible to restore from the backup, and then to roll the table space forward to the current point in time. In this case, all table space level backups taken prior to that time are no longer restorable. The restore operation will fail if the user tries to restore from such a backup. In cases where it cannot be determined that the backup is not valid (if, for instance, the database has been restored and rolled forward, thus creating a new log sequence), the restore may be successful, and the broken restore set will be detected during roll-forward recovery.

The user may choose to separate data, index, long field (LONG), and large objects (LOB) into different table spaces. Long field and LOB data for the same table must reside in the same table space.

Each component of a table may be backed up and restored with the table space in which it resides, independently of the other components of the table.

It is not necessary to back up table spaces for temporary tables. If a list of table spaces to be backed up contains such a table space, BACKUP fails.

Table space level backup and restore cannot be run concurrently.

### See Also

"MIGRATE DATABASE" on page 370

"RESTORE DATABASE" on page 441

"ROLLFORWARD DATABASE" on page 451.

## BIND

Invokes the bind utility, which prepares SQL statements stored in the bind file generated by the precompiler, and creates a package that is stored in the database.

### Scope

This command can be issued from any node in `db2nodes.cfg`. It updates the database catalogs on the catalog node. Its effects are visible to all nodes.

### Authorization

One of the following:

- *sysadm* or *dbadm* authority
- BINDADD privilege if a package does not exist and one of:
  - IMPLICIT_SCHEMA authority on the database if the schema name of the package does not exist
  - CREATEIN privilege on the schema if the schema name of the package exists
- ALTERIN privilege on the schema if the package exists
- BIND privilege on the package if it exists.

The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements. If the user has *sysadm* authority, but not explicit privileges to complete the bind, the database manager grants explicit *dbadm* authority automatically.

### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

### Command Syntax

**For DB2**

```
►►──BIND──filename──┬─────────────────────────────┬──┬───────────────────────────┬──────────►
                    │            ┌─UNAMBIG─┐      │  └─COLLECTION──schema-name──┘
                    └─BLOCKING───┼─ALL─────┼──────┘
                                 └─NO──────┘
```

```
►─┬────────────────────────┬──┬──────────────────────────────────────────┬──────────────►
  │            ┌─DEF─┐     │  │          ┌─1───────────────────────┐     │
  └─DATETIME───┼─EUR─┼─────┘  └─DEGREE───┼─degree-of-parallelism───┼─────┘
               ├─ISO─┤                   └─ANY─────────────────────┘
               ├─JIS─┤
               ├─LOC─┤
               └─USA─┘

►─┬──────────────────────────────┬──┬─────────────────────┬──┬──────────────────────┬──►
  │                ┌─RUN──┐      │  │          ┌─NO──┐    │  │           ┌─NO──┐    │
  └─DYNAMICRULES───┴─BIND─┴──────┘  └─EXPLAIN──┼─ALL─┼────┘  └─EXPLSNAP──┼─ALL─┼────┘
                                               └─YES─┘                   └─YES─┘

►─┬─────────────────────────────────┬──┬──────────────────────┬───────────────────────►
  │               ┌─,─────────────┐ │  └─GRANT───┬─authid─┬───┘
  │               ▼               │ │            └─PUBLIC─┘
  └─FUNCPATH──────schema-name─────┴─┘

►─┬──────────────────────────┬──┬────────────────────────┬──┬────────────────────┬────►
  └─GRANT_GROUP──group-name──┘  └─GRANT_USER──user-name──┘  │          ┌─DEF─┐   │
                                                            └─INSERT───┴─BUF─┴───┘

►─┬──────────────────────┬──┬────────────────────────────┬────────────────────────────►
  │             ┌─CS─┐   │  └─MESSAGES──message-file─────┘
  └─ISOLATION───┼─RR─┼───┘
                ├─RS─┤
                └─UR─┘

►─┬──────────────────────────────┬──┬──────────────────────────────┬──────────────────►
  └─OWNER──authorization-id──────┘  └─QUALIFIER──qualifier-name────┘

►─┬────────────────────────────────────┬──────────────────────────────────────────────►
  └─QUERYOPT──optimization-level───────┘

►──┬──────────────────────────────┬──┬──────────────────────┬─────────────────────────►◄
   │           ┌─NOPACKAGE─┐      │  │          ┌─YES─┐     │
   └─SQLERROR──┴─CHECK─────┴──────┘  └─SQLWARN──┴─NO──┴─────┘
```

## For DRDA

```
►►──BIND──filename────────────────────────────────────────────────────────────────────►
```

```
►──┬─────────────────────────────────────────────────────────────────────┬──────►
   └─ACTION──┬─ADD──────────────────────────────────────────────────────┤
             │ ┌─REPLACE─┐                                               │
             └─┴─────────┴──┬────────────────────────┬─┬───────────────────────┬┘
                            │          ┌─YES─┐       │ └─REPLVER──version-id─┘
                            └─RETAIN───┴─NO──┴───────┘

►──┬──────────────────────────┬─┬─────────────────────────┬─┬───────────────────────┬──►
   │          ┌─UNAMBIG─┐     │ └─CCSIDG──double-ccsid─┘ └─CCSIDM──mixed-ccsid─┘
   └─BLOCKING─┼─ALL─────┼─────┘
              └─NO──────┘

►──┬─────────────────────┬─┬──────────────────────┬─┬─────────────────────┬──────────►
   └─CCSIDS──sbcs-ccsid─┘ │          ┌─DEFAULT─┐ │ │           ┌─YES─┐   │
                           └─CHARSUB──┼─BIT─────┼─┘ └─CNULREQD──┴─NO──┴───┘
                                      ├─MIXED───┤
                                      └─SBCS────┘

►──┬───────────────────────────┬─┬──────────────────────────┬─┬─────────────┬──────►
   └─COLLECTION──schema-name─┘ │  (1)          ┌─DEF─┐    │ └─DEC──┬─15─┬─┘
                                └─────DATETIME──┼─EUR─┼────┘        └─31─┘
                                                ├─ISO─┤
                                                ├─JIS─┤
                                                ├─LOC─┤
                                                └─USA─┘

►──┬────────────────────────┬─┬─────────────────────────────────────────────┬──────►
   │         ┌─PERIOD─┐     │ │  (2)        ┌─1──────────────────────┐     │
   └─DECDEL──┴─COMMA──┴─────┘ └─────DEGREE──┼─degree-of-parallelism──┼─────┘
                                            └─ANY────────────────────┘

►──┬────────────────────────────────┬─┬─────────────────────────┬─┬───────────────────┬──►
   │               ┌─RUN────┐       │ │  (3)         ┌─NO──┐    │ └─GENERIC──string─┘
   └─DYNAMICRULES──┼─BIND───┼───────┘ └─────EXPLAIN──┴─YES─┴────┘
                   ├─DEFINE─┤
                   └─INVOKE─┘

►──┬───────────────────────┬─┬─────────────────────────────┬─────────────────────►
   └─GRANT──┬─authid─┬────┘ └─GRANT_GROUP──group-name────┘
            └─PUBLIC─┘

►──┬───────────────────────────┬─┬──────────────────────┬─┬───────────────────────┬──►
   └─GRANT_USER──user-name────┘ │         ┌─DEF─┐      │ │              ┌─CS─┐   │
                                 └─INSERT──┴─BUF─┴──────┘ └─ISOLATION────┼─NC─┼───┘
                                                                         ├─RR─┤
                                                                         ├─RS─┤
                                                                         └─UR─┘
```

```
►─┬──────────────────────────┬──┬──────────────────────────────┬──────────────────────────────►
  └─MESSAGES──message-file───┘  └─OWNER──authorization-id──────┘

►─┬──────────────────────────────┬──┬────────────────────────────────┬───────────────────────────►
  └─QUALIFIER──qualifier-name────┘  │              ┌─COMMIT──────┐    │
                                     └─RELEASE──────┴─DEALLOCATE──┴────┘

►─┬──────────────────────────────────┬──┬──────────────────────────────────┬──┬──────────────┬──►
  │                ┌─NOPACKAGE──┐    │  │             ┌─APOSTROPHE──┐      │  └─TEXT──label──┘
  └─SQLERROR───────┼─CHECK──────┼────┘  └─STRDEL──────┴─QUOTE───────┴──────┘
                   └─CONTINUE───┘

►─┬─────────────────────────────┬──────────────────────────────────────────────────────────────►◄
  │               ┌─RUN───┐     │
  └─VALIDATE──────┴─BIND──┴─────┘
```

**Notes:**

1. The DATETIME DEF option is not supported by DRDA, and is mapped to ISO when going through DB2 Connect.
2. The DEGREE option is only supported by DRDA Level 2 Application Servers.
3. DRDA defines the EXPLAIN option to have the value YES or NO. If the EXPLAIN option is not specified, this maps to DRDA ″EXPLAIN NO″.

## Command Parameters

**filename**

Specifies the name of the bind file that was generated when the application program was precompiled, or a list file containing the names of several bind files. Bind files have the extension `.bnd`. The full path name can be specified.

If a list file is specified, the @ character must be the first character of the list file name. The list file can contain several lines of bind file names. Bind files listed on the same line must be separated by plus (+) characters, but a + cannot appear in front of the first file listed on each line, or after the last bind file listed. For example,

```
/u/smith/sqllib/bnd/@all.lst
```

is a list file that contains the following bind files:

```
mybind1.bnd+mybind.bnd2+mybind3.bnd+
mybind4.bnd+mybind5.bnd+
mybind6.bnd+
mybind7.bnd
```

**ACTION**
Indicates whether the package can be added or replaced. This DRDA precompile/bind option is not supported by DB2.

**ADD** Indicates that the named package does not exist, and that a new package is to be created. If the package already exists, execution stops, and a diagnostic error message is returned.

**REPLACE**
Indicates that the old package is to be replaced by a new one with the same location, collection, and package name.

**RETAIN**
Indicates whether EXECUTE authorities are to be preserved when a package is replaced. If ownership of the package changes, the new owner grants the BIND and EXECUTE authority to the previous package owner.

**NO** Does not preserve EXECUTE authorities when a package is replaced.

**YES** Preserves EXECUTE authorities when a package is replaced.

**REPLVER version-id**
Replaces a specific version of a package. The version identifier specifies which version of the package is to be replaced. Maximum length is 254 characters.

**BLOCKING**
For information about row blocking, see the *Administration Guide* or the *Application Development Guide.*

**ALL** Specifies to block for:

- Read-only cursors
- Cursors not specified as FOR UPDATE OF

Ambiguous cursors are treated as read-only.

**NO** Specifies not to block any cursors. Ambiguous cursors are treated as updatable.

**UNAMBIG**
Specifies to block for:

- Read-only cursors
- Cursors not specified as FOR UPDATE OF

Ambiguous cursors are treated as updatable.

**CCSIDG double-ccsid**
An integer specifying the coded character set identifier (CCSID) to be used for double byte characters in character column definitions (without a specific CCSID clause) in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**CCSIDM mixed-ccsid**
An integer specifying the coded character set identifier (CCSID) to be used for mixed byte characters in character column definitions (without a specific CCSID clause) in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**CCSIDS sbcs-ccsid**
An integer specifying the coded character set identifier (CCSID) to be used for single byte characters in character column definitions (without a specific CCSID clause) in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**CHARSUB**
Designates the default character sub-type that is to be used for column definitions in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2.

**BIT** Use the FOR BIT DATA SQL character sub-type in all new character columns for which an explicit sub-type is not specified.

**DEFAULT**
Use the target system defined default in all new character columns for which an explicit sub-type is not specified.

**MIXED**
Use the FOR MIXED DATA SQL character sub-type in all new character columns for which an explicit sub-type is not specified.

**SBCS** Use the FOR SBCS DATA SQL character sub-type in all new character columns for which an explicit sub-type is not specified.

**CNULREQD**
This option is related to the **langlevel** precompile option, which is not

supported by DRDA. It is valid only if the bind file is created from a C or a C++ application. This DRDA bind option is not supported by DB2.

**NO** The application was coded on the basis of the **langlevel** SAA1 precompile option with respect to the null terminator in C string host variables.

**YES** The application was coded on the basis of the **langlevel** MIA precompile option with respect to the null terminator in C string host variables.

**COLLECTION schema-name**
Specifies an 8-character collection identifier for the package. If not specified, the authorization identifier for the user processing the package is used.

**DATETIME**
Specifies the date and time format to be used. For more information about date and time formats, see the *SQL Reference*.

**DEF** Use a date and time format associated with the country code of the database.

**EUR** Use the IBM standard for Europe date and time format.

**ISO** Use the date and time format of the International Standards Organization.

**JIS** Use the date and time format of the Japanese Industrial Standard.

**LOC** Use the date and time format in local form associated with the country code of the database.

**USA** Use the IBM standard for U.S. date and time format.

**DEC** Specifies the maximum precision to be used in decimal arithmetic operations. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**15** 15-digit precision is used in decimal arithmetic operations.

**31** 31-digit precision is used in decimal arithmetic operations.

**DECDEL**
Designates whether a period (.) or a comma (,) will be used as the decimal point indicator in decimal and floating point literals. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**COMMA**
Use a comma (,) as the decimal point indicator.

**PERIOD**
Use a period (.) as the decimal point indicator.

**DEGREE**
Specifies the degree of parallelism for the execution of static SQL statements in an SMP system. This option does not affect CREATE INDEX parallelism.

**1** The execution of the statement will not use parallelism.

**degree-of-parallelism**
Specifies the degree of parallelism with which the statement can be executed, a value between 2 and 32 767 (inclusive).

**ANY** Specifies that the execution of the statement can involve parallelism using a degree determined by the database manager.

**DYNAMICRULES**
Defines which rules apply to dynamic SQL at run time for the initial setting of the values used for authorization ID and for the implicit qualification of unqualified object references.

**RUN** Specifies that the authorization ID of the user executing the package is to be used. This is the default value.

**BIND** Specifies that all of the rules that apply to static SQL for authorization and qualification are to be used at run time. That is, the authorization ID of the package owner is to be used for authorization checking of dynamic SQL statements, and the default package qualifier is to be used for implicit qualification of unqualified object references within dynamic SQL statements.

When binding a package with this option, the binder of the package should not have any authorities that the user of the package should not receive, because dynamic SQL statements will be using the authorization ID of the package owner. The following dynamically prepared SQL statements cannot be used within a package that has been bound with this option: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET CONSTRAINTS, and SET EVENT MONITOR STATE.

**DEFINE**
Indicates that the authorization identifier used for the execution of dynamic SQL statements in a UDF or stored

procedure is the definer of the UDF or stored procedure. This option is not supported by DB2.

**INVOKE**
Indicates that the authorization identifier used for the execution of dynamic SQL statements in a UDF or stored procedure is the invoker of the UDF or stored procedure. This option is not supported by DB2.

**EXPLAIN**
Stores information in the Explain tables about the access plans chosen for each SQL statement in the package. DRDA does not support the ALL value for this option.

**NO** Explain information will not be captured.

**YES** Explain tables will be populated with information about the chosen access plan.

**ALL** Explain information for each eligible static SQL statement will be placed in the Explain tables. In addition, Explain information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to `NO`. For more information about special registers, see the *SQL Reference*.

**Note:** This value for EXPLAIN is not supported by DRDA.

**EXPLSNAP**
Stores Explain Snapshot information in the Explain tables. This DB2 precompile/bind option is not supported by DRDA.

**NO** An Explain Snapshot will not be captured.

**YES** An Explain Snapshot for each eligible static SQL statement will be placed in the Explain tables.

**ALL** An Explain Snapshot for each eligible static SQL statement will be placed in the Explain tables. In addition, Explain Snapshot information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to `NO`. For more information about special registers, see the *SQL Reference*.

**FUNCPATH**
Specifies the function path to be used in resolving user-defined distinct types and functions in static SQL. If this option is not specified, the default function path is "SYSIBM","SYSFUN",USER where USER is the value of the USER special register. This DB2 precompile/bind option is not supported by DRDA.

**schema-name**
A short SQL identifier, either ordinary or delimited, which identifies a schema that exists at the application server. No validation that the schema exists is made at precompile or at bind time. The same schema cannot appear more than once in the function path. The number of schemas that can be specified is limited by the length of the resulting function path, which cannot exceed 254 bytes. The schema SYSIBM does not need to be explicitly specified; it is implicitly assumed to be the first schema if it is not included in the function path. For more information, see the *SQL Reference*.

**GENERIC string**
Provides a means of passing new bind options to a target DRDA database. Supports the binding of new options that are defined in the target database, but that are not known to the local command. Do not use this option to pass bind options that *are* defined in "BIND" on page 131 or "PRECOMPILE PROGRAM" on page 372. This option can substantially improve dynamic SQL performance. The syntax is as follows:

```
generic "option1 value1 option2 value2 ..."
```

Each option and value must be separated by one or more blank spaces. For example, if the target DRDA database is DB2 MVS Version 5, one could use:

```
generic "keepdynamic yes"
```

to bind the new **keepdynamic** YES option, which is not defined locally on the PRECOMPILE PROGRAM or the BIND command.

The maximum length of the string is 1023 bytes. This DRDA bind option is currently only supported by DB2 MVS Version 5; it is not supported by DB2.

**GRANT**

**authid** Grants EXECUTE and BIND privileges to a specified user name or group ID.

**PUBLIC**
Grants EXECUTE and BIND privileges to PUBLIC.

**GRANT_GROUP group-name**
Grants EXECUTE and BIND privileges to a specified group ID.

**GRANT_USER user-name**
Grants EXECUTE and BIND privileges to a specified user name.

**INSERT**
: Allows a program being precompiled or bound against a DB2 Universal Database Extended Enterprise Edition server to request that data inserts be buffered to increase performance.

| | |
|---|---|
| **BUF** | Specifies that inserts from an application should be buffered. |
| **DEF** | Specifies that inserts from an application should not be buffered. |

**ISOLATION**
: Determines how far a program bound to this package can be isolated from the effect of other executing programs. For more information about isolation levels, see the *SQL Reference*.

| | |
|---|---|
| **CS** | Specifies Cursor Stability as the isolation level. |
| **NC** | No Commit. Specifies that commitment control is not to be used. This isolation level is not supported by DB2. |
| **RR** | Specifies Repeatable Read as the isolation level. |
| **RS** | Specifies Read Stability as the isolation level. Read Stability ensures that the execution of SQL statements in the package is isolated from other application processes for rows read and changed by the application. |
| **UR** | Specifies Uncommitted Read as the isolation level. |

**MESSAGES message-file**
: Specifies the destination for warning, error, and completion status messages. A message file is created whether the bind is successful or not. If a message file name is not specified, the messages are written to standard output. If the complete path to the file is not specified, the current directory is used. If the name of an existing file is specified, the contents of the file are overwritten.

**OWNER authorization-id**
: Designates an 8-character authorization identifier for the package owner. The owner must have the privileges required to execute the SQL statements contained in the package. Only a user with SYSADM or DBADM authority can specify an authorization identifier other than the user ID. The default value is the primary authorization ID of the precompile/bind process. SYSIBM, SYSCAT, and SYSSTAT are not valid values for this option.

**QUALIFIER qualifier-name**
: Provides an 8-character implicit qualifier for unqualified objects contained in the package. The default is the owner's authorization ID, whether or not **owner** is explicitly specified.

**QUERYOPT optimization-level**
Indicates the desired level of optimization for all static SQL statements contained in the package. The default value is 5. For the complete range of optimization levels available, see the SET CURRENT QUERY OPTIMIZATION statement in the *SQL Reference*. This DB2 precompile/bind option is not supported by DRDA.

**RELEASE**
Indicates whether resources are released at each COMMIT point, or when the application terminates. This DRDA precompile/bind option is not supported by DB2.

**COMMIT**
Release resources at each COMMIT point. Used for dynamic SQL statements.

**DEALLOCATE**
Release resources only when the application terminates.

**SQLERROR**
Indicates whether to create a package or a bind file if an error is encountered.

**CHECK**
Specifies that the target system performs all syntax and semantic checks on the SQL statements being bound. A package will not be created as part of this process. If, while binding, an existing package with the same name and version is encountered, the existing package is neither dropped nor replaced even if **action replace** was specified.

**CONTINUE**
A package or a bind file is created even when SQL errors are encountered. This option is not supported by DB2.

**NOPACKAGE**
A package or a bind file is not created if an error is encountered.

**SQLWARN**
Indicates whether warnings will be returned from the compilation of dynamic SQL statements (via PREPARE or EXECUTE IMMEDIATE), or from describe processing (via PREPARE...INTO or DESCRIBE). This DB2 precompile/bind option is not supported by DRDA.

**NO** Warnings will not be returned from the SQL compiler.

**YES** Warnings will be returned from the SQL compiler.

**Note:** SQLCODE +238 is an exception. It is returned regardless of the **sqlwarn** option value.

**STRDEL**

Designates whether an apostrophe (’) or double quotation marks (") will be used as the string delimiter within SQL statements. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**APOSTROPHE**

Use an apostrophe (’) as the string delimiter.

**QUOTE**

Use double quotation marks (") as the string delimiter.

**TEXT label**

The description of a package. Maximum length is 255 characters. The default value is blanks. This DRDA precompile/bind option is not supported by DB2.

**VALIDATE**

Determines when the database manager checks for authorization errors and object not found errors. The package owner authorization ID is used for validity checking. This DRDA precompile/bind option is not supported by DB2.

**BIND** Validation is performed at precompile/bind time. If all objects do not exist, or all authority is not held, error messages are produced. If **sqlerror continue** is specified, a package/bind file is produced despite the error message, but the statements in error are not executable.

**RUN** Validation is attempted at bind time. If all objects exist, and all authority is held, no further checking is performed at execution time.

If all objects do not exist, or all authority is not held at precompile/bind time, warning messages are produced, and the package is successfully bound, regardless of the **sqlerror continue** option setting. However, authority checking and existence checking for SQL statements that failed these checks during the precompile/bind process may be redone at execution time.

## Examples

The following example binds `myapp.bnd` (the bind file generated when the `myapp.sqc` program was precompiled) to the database to which a connection has been established:

```
db2 bind myapp.bnd
```

Any messages resulting from the bind process are sent to standard output.

## Usage Notes

Binding can be done as part of the precompile process for an application program source file, or as a separate step at a later time. Use BIND when binding is performed as a separate process.

The name used to create the package is stored in the bind file, and is based on the source file name from which it was generated (existing paths or extensions are discarded). For example, a precompiled source file called `myapp.sql` generates a default bind file called `myapp.bnd` and a default package name of `MYAPP`. However, the bind file name and the package name can be overridden at precompile time by using the **bindfile** and the **package** options.

Binding a package with a schema name that does not already exist results in the implicit creation of that schema. The schema owner is SYSIBM. The CREATEIN privilege on the schema is granted to PUBLIC.

BIND executes under the transaction that was started. After performing the bind, BIND issues a COMMIT or a ROLLBACK to terminate the current transaction and start another one.

Binding stops if a fatal error or more than 100 errors occur. If a fatal error occurs, the utility stops binding, attempts to close all files, and discards the package.

If a package is bound with **dynamicrules bind**, the implicit or explicit value of the bind option **owner** is used for authorization checking of dynamic SQL statements, and the implicit or explicit value of the bind option **qualifier** is used as the implicit qualifier of unqualified objects within dynamic SQL statements. If multiple packages are referenced during a single connection, dynamic SQL statements prepared by a specific package will behave according to the bind options for that package. The value of the special register CURRENT SCHEMA has no effect on qualification in a package bound with **dynamicrules bind**.

Binding application programs has prerequisite requirements and restrictions beyond the scope of this manual. For more detailed information about binding application programs to databases, see the *Application Development Guide*.

## See Also

"PRECOMPILE PROGRAM" on page 372.

## CATALOG APPC NODE

Adds an APPC node entry to the node directory. The Advanced Program-to-Program Communications protocol is used to access the remote node.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CATALOG──┬───────┬──APPC NODE──nodename──REMOTE──symbolic-destination-name──────────►
             └─ADMIN─┘

►──┬───────────────────────────────┬──┬────────────────────────────────┬────────────────►
   │               ┌─PROGRAM─┐     │  └─REMOTE_INSTANCE──instance-name─┘
   └─SECURITY──────┼─NONE────┼─────┘
                   └─SAME────┘

►──┬──────────────────────────┬──┬─────────────────────────────────────┬────────────────►
   └─SYSTEM──system-name──────┘  └─OSTYPE──operating-system-type───────┘

►──┬──────────────────────────────┬─────────────────────────────────────────────────────►◄
   └─WITH──"comment-string"───────┘
```

### Command Parameters

**ADMIN**
Specifies administration server nodes.

**NODE nodename**
A local alias for the node to be cataloged. This is an arbitrary name on the user's workstation, used to identify the node. It should be a meaningful name to make it easier to remember. The name must conform to database manager naming conventions (see "Appendix B. Naming Conventions" on page 525).

**REMOTE symbolic-destination-name**
Specifies the symbolic destination name of the remote partner node.

The name corresponds to an entry in the CPI Communications side information table that contains the necessary information for the client to set up an APPC connection to the server (partner LU name, mode name, partner TP name). Maximum length is 8 characters.

**SECURITY**

**PROGRAM**
Specifies that both a user name and a password are to be included in the allocation request sent to the partner LU.

**NONE**
Specifies that no security information is to be included in the allocation request sent to the partner LU.

**SAME** Specifies that a user name is to be included in the allocation request sent to the partner LU, together with an indicator that the user name has been ″already verified″. The partner must be configured to accept ″already verified″ security.

**REMOTE_INSTANCE instance-name**
Specifies the real name of the instance to which an attachment is being made on the remote server machine.

**SYSTEM system-name**
Specifies a name that is used to identify the server machine.

**OSTYPE operating-system-type**
Specifies the operating system type of the server machine. Valid values are: OS2, AIX, WIN95, NT, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, and SGI.

**WITH ″comment-string″**
Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Examples

```
db2 catalog appc node db2appc1 remote db2inst1 security program
   with "A remote APPC node"
```

## Usage Notes

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). On an OS/2 client or a Windows client, it stores and maintains the node directory in the instance subdirectory where the client is installed. On an AIX client, it creates the node directory in the DB2 installation directory.

List the contents of the local node directory using "LIST NODE DIRECTORY" on page 319.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in "GET DATABASE MANAGER CONFIGURATION" on page 236), database, node, and DCS directory files are cached in memory. An application's directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP's directory cache, use "TERMINATE" on page 484. To refresh DB2's shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CATALOG APPCLU NODE

Writes information to the node directory about a remote workstation that uses APPC as its communication protocol. DB2 uses this information to establish the connection between an application and a remote database cataloged on this node.

This command is available on OS/2 only.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CATALOG APPCLU NODE──nodename──┬─────────────────────┬──REMOTE──partner-lu──────────────►
                                   └─LOCAL──local-lu──┘

►──┬─────────────┬──┬──────────────────────────┬─────────────────────────────────────────►◄
   └─MODE──mode──┘  └─WITH──"comment-string"──┘
```

### Command Parameters

**NODE nodename**
: Specifies the name of the remote workstation to catalog. This is the same name that is entered for the node name parameter when cataloging a database residing on that workstation. The name must conform to DB2 naming conventions (see “Appendix B. Naming Conventions” on page 525).

**LOCAL local-lu**
: Specifies the alias of the SNA local logical unit used for the connection. It must be a string containing 1 to 8 non-blank characters. The alias must be entered exactly as it appears (using mixed case characters) in the corresponding SNA definition (from the Communication Manager configuration).

**REMOTE partner-lu**
: Specifies the alias of the SNA remote logical unit used for the connection. It must be a string containing 1 to 8 non-blank characters. The alias must be entered exactly as it appears (using mixed case

characters) in the corresponding SNA definition (from the Communication Manager configuration).

**MODE mode**
Specifies the SNA transmission mode used for the connection. The name must conform to SNA naming conventions.

If a value is not entered, DB2 stores a character string of eight blanks as the mode type.

**WITH ″comment-string″**
Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Examples

The following example shows how to catalog REMNODE, a remote workstation that contains a database:

```
db2 catalog appclu node remnode local LU1 remote LU2
   mode SQLMOD01 with "Remote node comment"
```

## Usage Notes

This command is identical to the Version 1 CATALOG APPC NODE command, but is different from the Version 2 CATALOG APPC NODE command.

**Note:** Version 1 script files containing the CATALOG APPC NODE command must be modified. Change the keyword APPC to APPCLU to make this command perform as it did in Version 1.

“ATTACH” on page 121 cannot be used with an APPCLU node to attach to a server that has TPNAME other than x’07F6C4C2’.

APPCLU nodes only support security PROGRAM (security SAME and NONE are not supported).

## CATALOG APPN NODE

Writes information to the node directory about a remote workstation that uses APPN as its communication protocol. DB2 uses this information to establish the connection between an application and a remote database cataloged on this node.

This command is available on OS/2, Windows NT, Windows 98, and Windows 95 only.

### Authorization

*sysadm*

### Required Connection

None

### Command Syntax

```
►►──CATALOG APPN NODE──nodename──┬────────────────────┬──REMOTE──partner-lu──────────►
                                 └─NETWORKID──netid──┘

►──┬──────────────────┬──┬───────────────┬──┬─────────────────────────┬──────────────►
   └─LOCAL──local-lu──┘  └─MODE──mode──┘  └─WITH──"comment-string"──┘

►──┬──────────────────┬──┬──────────────────────────┬────────────────────────────────►
   └─TPNAME──tpname──┘  └─SECURITY──┬─NONE────┬──┘
                                    ├─SAME────┤
                                    └─PROGRAM─┘

►──┬────────────────────────────────────┬────────────────────────────────────────────►◄
   └─CHGPWDLU──change_password_lu──┘
```

### Command Parameters

**NODE nodename**
: Specifies the name of the remote workstation to catalog. This is the same name that is entered for the node name parameter when a database residing on that workstation is cataloged (using "CATALOG DATABASE" on page 153). The name must conform to DB2 naming conventions (see "Appendix B. Naming Conventions" on page 525).

**NETWORKID netid**
: Specifies the ID of the SNA network where the remote LU resides. This network ID is a string of one to eight characters that follows naming conventions for SNA.

**REMOTE partner-lu**
: Specifies the SNA partner logical unit used for the connection. Enter the LU name of the remote node. The name must be entered exactly as it appears (using mixed case characters) in the corresponding SNA definition (from the Communication Manager configuration). The name must follow SNA naming conventions.

**LOCAL local-lu**
: Specifies the alias of the SNA local logical unit used for the connection. It must be a string containing 1 to 8 non-blank characters. The alias must be entered exactly as it appears (using mixed case characters) in the corresponding SNA definition (from the Communication Manager configuration).

**MODE mode**
: Specifies the SNA transmission mode used for the connection. The name must conform to SNA naming conventions.

  If a value is not entered, DB2 stores a character string of eight blanks as the mode type.

**WITH ″comment-string″**
: Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

**TPNAME tpname**
: Specifies the APPC transaction program name of the database server.

**SECURITY**
: Specifies the APPC security level. Valid values are:

  **NONE**
  : Specify this value for a DB2 UDB server.

  **SAME**

  **PROGRAM**
  : Specify this value for a host database server.

**CHGPWDLU change_password_lu**
: Specifies the name of the partner LU that is to be used when changing the password for a host database server.

## Examples

The following example catalogs an APPN node:

```
db2 catalog appn node remnode remote rlu with "Catalog APPN NODE"
```

## Usage Notes

The *tpname* must be an application TP (for example, `DB2DRDA`). If not specified, the default is a service TP (`0x07F6C4C2`).

## CATALOG DATABASE

Stores database location information in the system database directory. The database can be located either on the local workstation or on a remote node.

### Scope

In a partitioned database environment, when cataloging a local database into the system database directory, this command must be issued from a node on the server where the database resides.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

None. Directory operations affect the local directory only.

### Command Syntax

```
>>──CATALOG──┬─DATABASE─┬──database-name──┬────────────┬──┬───────────────────────┬──────────>
             └─DB───────┘                  └─AS──alias──┘  ├─ON──┬─path──┬─────────┤
                                                            │     └─drive─┘         │
                                                            └─AT NODE──nodename─────┘

>──┬─────────────────────────────────────────────────────────────────┬────────────────────>
   └─AUTHENTICATION──┬─SERVER────────────────────────────────────┬──┘
                     ├─CLIENT────────────────────────────────────┤
                     ├─DCS───────────────────────────────────────┤
                     ├─SERVER_ENCRYPT────────────────────────────┤
                     ├─DCS_ENCRYPT───────────────────────────────┤
                     └─DCE SERVER PRINCIPAL──principalname───────┘

>──┬───────────────────────────┬──────────────────────────────────────────────────────────><
   └─WITH──"comment-string"────┘
```

### Command Parameters

**DATABASE database-name**
Specifies the name of the database to catalog.

**AS alias**
Specifies an alias as an alternate name for the database being cataloged. If an alias is not specified, the database manager uses *database-name* as the alias.

**ON path/drive**
On UNIX based systems, specifies the path on which the database being cataloged resides. On OS/2 or the Windows operating system, specifies the letter of the drive on which the database being cataloged resides.

**AT NODE nodename**
Specifies the name of the node where the database being cataloged resides. This name should match the name of an entry in the node directory. If the node name specified does not exist in the node directory, a warning is returned, but the database is cataloged in the system database directory. The node name should be cataloged in the node directory if a connection to the cataloged database is desired.

**AUTHENTICATION**

**Note:** Required only if a DB2 Version 2 or greater client is communicating with a DB2 Version 1 server.

The authentication value is stored for remote databases (it appears in the output from "LIST DATABASE DIRECTORY" on page 299) but it is not stored for local databases.

Specifying an authentication type can result in a performance benefit. For more information about authentication types, see the *Administration Guide*.

**SERVER**
Specifies that authentication takes place on the node containing the target database.

**CLIENT**
Specifies that authentication takes place on the node where the application is invoked.

**DCS** Specifies that authentication takes place on the node containing the target database, except when using DB2 Connect, when it specifies that authentication takes place at the DRDA application server (AS).

**SERVER_ENCRYPT**
Specifies that authentication takes place on the node containing the target database, and that passwords are

encrypted at the source. Passwords are decrypted at the target, as specified by the authentication type cataloged at the source.

**DCS_ENCRYPT**
Specifies that authentication takes place on the node containing the target database, except when using DB2 Connect; in that case, authentication takes place at the DRDA application server (AS). Passwords are encrypted at the source, and decrypted at the target, as specified by the authentication type cataloged at the source.

**DCE** Specifies that authentication takes place using DCE Security Services. When authentication is DCE, and an APPC connection is used for access, only SECURITY=NONE is supported.

**SERVER PRINCIPAL principalname**
Fully qualified DCE principal name for the target server. This value is also recorded in the keytab file at the target server.

**WITH ″comment-string″**
Describes the database or the database entry in the system database directory. The maximum length of a comment string is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

## Examples

```
db2 catalog database sample on /databases/sample
   with "Sample Database"
```

## Usage Notes

Use CATALOG DATABASE to catalog databases located on local or remote nodes, recatalog databases that were uncataloged previously, or maintain multiple aliases for one database (regardless of database location).

DB2 automatically catalogs databases when they are created. It catalogs an entry for the database in the local database directory and another entry in the system database directory. If the database is created from a remote client (or a client which is executing from a different instance on the same machine), an entry is also made in the system database directory at the client instance.

If neither path nor node name is specified, the database is assumed to be local, and the location of the database is assumed to be that specified in the database manager configuration parameter *dftdbpath*.

Databases on the same node as the database manager instance are cataloged as *indirect* entries. Databases on other nodes are cataloged as *remote* entries.

CATALOG DATABASE automatically creates a system database directory if one does not exist. The system database directory is stored on the path that contains the database manager instance that is being used, and is maintained outside of the database.

List the contents of the system database directory using “LIST DATABASE DIRECTORY” on page 299.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## See Also

“UNCATALOG DATABASE” on page 485.

## CATALOG DCS DATABASE

Stores information about remote databases in the Database Connection Services (DCS) directory. These databases are accessed through an Application Requester (AR), such as DB2 Connect. Having a DCS directory entry with a database name matching a database name in the system database directory invokes the specified AR to forward SQL requests to the remote server where the database resides. For more information about DB2 Connect and DCS directory entries, see the *DB2 Connect User's Guide*.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CATALOG DCS──┬─DATABASE─┬──database-name──┬──────────────────────────┬──────────►
                 └─DB───────┘                 └─AS──target-database-name─┘

►──┬──────────────────┬──┬──────────────────────────────┬─────────────────────────►
   └─AR──library-name─┘  └─PARMS──"parameter-string"──┘

►──┬────────────────────────────┬─────────────────────────────────────────────────►◄
   └─WITH──"comment-string"──┘
```

### Command Parameters

**DATABASE database-name**
: Specifies the alias of the target database to catalog. This alias must match the database name entered during the remote cataloging of this database in the system database directory.

**AS target-database-name**
: Specifies the name of the target host database to catalog.

**AR library-name**
: Specifies the name of the Application Requester library that is loaded and used to access a remote database listed in the DCS directory.

**Note:** If using the DB2 Connect AR, do not specify a library name. The default value will cause DB2 Connect to be invoked.

If not using DB2 Connect, specify the library name of the AR, and place that library on the same path as the database manager libraries. On OS/2 or the Windows operating system, the path is `drive:\sqllib\dll`. On UNIX based systems, the path is `$HOME/sqllib/lib` of the instance owner.

**PARMS** ″**parameter-string**″
Specifies a parameter string that is to be passed to the AR when it is invoked. The parameter string must be enclosed by double quotation marks.

**WITH** ″**comment-string**″
Describes the DCS directory entry. Any comment that helps to describe the database cataloged in this directory can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

## Examples

The following example catalogs information about the DB1 database, which is a DB2 for MVS host database, into the DCS directory:

```
db2 catalog dcs database db1 as dsn_db_1
   with "DB2/MVS location name DSN_DB_1"
```

## Usage Notes

The DB2 Connect program provides connections to DRDA Application Servers such as:

- DB2 for OS/390 databases on System/370 and System/390 architecture host computers
- DB2 for VM and VSE databases on System/370 and System/390 architecture host computers
- OS/400 databases on Application System/400 (AS/400) host computers.

The database manager creates a Database Connection Services directory if one does not exist. This directory is stored on the path that contains the database manager instance that is being used. The DCS directory is maintained outside of the database.

The database must also be cataloged as a remote database in the system database directory.

List the contents of the DCS directory using “LIST DCS DIRECTORY” on page 306 .

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## See Also

“UNCATALOG DCS DATABASE” on page 487.

## CATALOG GLOBAL DATABASE

Creates a system database directory entry of the DCE type. This entry is used to define a local alias of the fully qualified DCE directory object name of the target database. The information about that database is stored centrally in the DCE directory.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CATALOG GLOBAL──┬─DATABASE─┬──database-global-name─────────────────────────────►
                    └─DB───────┘

►─AS──alias──USING DIRECTORY──DCE──┬─────────────────────────┬──────────────────────►◄
                                   └─WITH──"comment-string"──┘
```

### Command Parameters

**DATABASE database-global-name**
: Specifies the fully qualified name that uniquely identifies the database in the DCE name space.

**AS alias**
: Specifies an alternate name for the database being cataloged.

**USING DIRECTORY DCE**
: Specifies the global directory service being used.

**WITH "comment-string"**
: Describes the DCE type entry in the system database directory. Any comment that helps to describe the database cataloged in this directory can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

### Examples

```
db2 catalog global database /.../cell1/subsys/database/DB3
   as dbtest using directory dce
```

## Usage Notes

The maximum length of *database-global-name* is 255 bytes. The name must begin with either `/.../` or `/.:/`.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in "GET DATABASE MANAGER CONFIGURATION" on page 236), database, node, and DCS directory files are cached in memory. An application's directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP's directory cache, use "TERMINATE" on page 484. To refresh DB2's shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CATALOG IPX/SPX NODE

Adds an Internetwork Packet Exchange/Sequenced Packet Exchange (IPX/SPX) node entry to the node directory. The Novell NetWare IPX/SPX communications protocol is used to access the remote node.

This command is available on OS/2, Windows NT, and Windows 95 only.

### Scope

IPX/SPX file server addressing is not supported in a multi-node environment.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CATALOG──┬───────┬──IPXSPX NODE──nodename──REMOTE──file-server──────────────────►
             └─ADMIN─┘

►─SERVER──object-name──┬────────────────────────────────────┬───────────────────────►
                       └─REMOTE_INSTANCE──instance-name──┘

►──┬──────────────────────┬──┬─────────────────────────────────────┬────────────────►
   └─SYSTEM──system-name──┘  └─OSTYPE──operating-system-type──┘

►──┬─────────────────────────┬───────────────────────────────────────────────────►◄
   └─WITH──"comment-string"──┘
```

### Command Parameters

**ADMIN**
: Specifies that an IPX/SPX administration server node is to be cataloged.

**NODE nodename**
: A local alias for the node to be cataloged. This is an arbitrary name on the user's workstation, used to identify the node. It should be a meaningful name to make it easier to remember. The name must

conform to database manager naming conventions (see “Appendix B. Naming Conventions” on page 525).

**REMOTE file-server**

Name of the NetWare file server where the internetwork address of the server database manager instance is registered. The internetwork address is stored in the bindery at the NetWare file server, and is accessed using *object-name*.

**Note:** The following characters are not valid: / \ : ; , * ?

**SERVER object-name**

Name of the database manager instance stored in the bindery of the NetWare file server. Each server database manager instance registered at one NetWare file server must be represented by a unique *object-name*. It is recommended that each database manager instance on the network be represented by a unique *object-name*.

**Note:** The following characters are not valid: / \ : ; , * ?

When cataloging the IPX/SPX client to use *file server* addressing, specify the file server and object name as defined above. When cataloging the IPX/SPX client to use *direct* addressing, specify *file-server* as *, and specify the server’s IPX/SPX internetwork address in the *object-name* parameter. Use “db2ipxad - Get IPX/SPX Internetwork Address” on page 48 to retrieve the server’s IPX/SPX internetwork address. For more information about the addressing methods, see one of the *Quick Beginnings* books.

**REMOTE_INSTANCE instance-name**

Specifies the name of the server instance to which an attachment is being made.

**SYSTEM system-name**

Specifies the DB2 system name that is used to identify the server machine.

**OSTYPE operating-system-type**

Specifies the operating system type of the server machine. Valid values are: OS2, AIX, WIN95, NT, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, and SGI.

**WITH ″comment-string″**

Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Examples

```
db2 catalog ipxspx node db2ipx1 remote netwsrv server db2inst1
   with "A remote IPX/SPX node"

db2 catalog ipxspx node db2ipx2 remote * server 09212700.400011527745.879E
   with "IPX/SPX node using direct addr"
```

## Usage Notes

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). On an OS/2 client or a Windows client, it stores and maintains the node directory in the instance subdirectory where the client is installed. On an AIX client, it creates the node directory in the DB2 installation directory.

List the contents of the local node directory using “LIST NODE DIRECTORY” on page 319.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CATALOG LDAP DATABASE

Used to register the database in LDAP (Lightweight Directory Access Protocol).

This command is available on Windows NT, Windows 98, and Windows 95 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──CATALOG LDAP──┬─DATABASE─┬──database-name──┬────────────┬──────────────────────────►
                  └─DB───────┘                 └─AS──alias──┘

►──┬─────────────────────┬──┬──────────────────────────┬──┬────────────────────────────────┬──►
   └─AT NODE──nodename───┘  └─GWNODE──gateway-node─────┘  └─PARMS──"parameter-string"──────┘

►──┬─────────────────────┬──┬──────────────────────────────────────────┬──────────────────►
   └─AR──library-name────┘  └─AUTHENTICATION──┬─CLIENT──────────┬──────┘
                                              ├─SERVER──────────┤
                                              ├─SERVER_ENCRYPT──┤
                                              ├─DCS_ENCRYPT─────┤
                                              └─DCS─────────────┘

►──┬─────────────────────┬──┬──────────────────────────────────────────────┬──────────────►◄
   └─WITH──"comments"────┘  └─USER──username──┬─────────────────────────┬──┘
                                              └─PASSWORD──password──────┘
```

### Command Parameters

**DATABASE database-name**
: Specifies the name of the database to catalog.

**AS alias**
: Specifies an alias as an alternate name for the database being cataloged. If an alias is not specified, the database name is used as the alias.

**AT NODE nodename**
Specifies the LDAP node name for the database server on which the database resides. This parameter must be specified when registering a database on a remote server.

**GWNODE gateway-node**
Specifies the LDAP node name for the gateway server.

**PARMS ″parameter-string″**
Specifies a parameter string that is passed to the Application Requester (AR) when accessing DCS databases. For a description of what format DDCS expects for this string, see the *DB2 Connect User's Guide*.

**Note:** The change password *sym_dest_name* should not be specified in the parameter string. Use the keyword CHGPWDLU to specify the change password LU name when registering the DB2 server in LDAP. For more information, see "REGISTER" on page 414.

**AR library-name**
Specifies the name of the Application Requester library that is loaded and used to access a remote database listed in the DCS directory.

**Note:** If using the DB2 Connect AR, do not specify a library name. The default value will cause DB2 Connect to be invoked.

If not using DB2 Connect, specify the library name of the AR, and place that library on the same path as the database manager libraries. On OS/2 or the Windows operating system, the path is `drive:\sqllib\dll`. On UNIX based systems, the path is `$HOME/sqllib/lib` of the instance owner.

**AUTHENTICATION**
Specifies the authentication level. For detailed information about authentication types, including performance implications, see the *Administration Guide*. Valid values are:

**CLIENT**
Specifies that authentication takes place on the node from which the application is invoked.

**SERVER**
Specifies that authentication takes place on the node containing the target database.

**SERVER_ENCRYPT**
Specifies that authentication takes place on the node containing the target database, and that passwords are

encrypted at the source. Passwords are decrypted at the target, as specified by the authentication type cataloged at the source.

**DCS_ENCRYPT**
Specifies that authentication takes place on the node containing the target database, except when using DB2 Connect; in that case, authentication takes place at the DRDA application server (AS). Passwords are encrypted at the source, and decrypted at the target, as specified by the authentication type cataloged at the source.

**DCS** Specifies that authentication takes place on the node containing the target database, except when using DB2 Connect; in that case, authentication takes place at the DRDA application server (AS).

**WITH ″comments″**
Describes the DB2 server. Any comment that helps to describe the server registered in the network directory can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

**USER username**
Specifies the user's LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create the object in the LDAP directory. If the user's LDAP DN is not specified, the credentials of the current logon user will be used.

**PASSWORD password**
Account password.

## Usage Notes

If the node name is not specified, DB2 will use the first node in LDAP that represents the DB2 server on the current machine.

Authentication type DCE is not supported.

It may be necessary to manually register (catalog) the database in LDAP if:

- The database server does not support LDAP. The administrator must manually register each database in LDAP to allow clients that support LDAP to access the database without having to catalog the database locally on each client machine.
- The application wants to use a different name to connect to the database. In this case, the administrator can catalog the database using a different alias name.

- The database resides at the host database server (for example, DB2/390, DB2/400, and so on). In this case, the administrator can register the database in LDAP and specify the gateway node through the GWNODE parameter.
- During CREATE DATABASE IN LDAP the database name already exists in LDAP. The database is still created on the local machine (and can be accessed by local applications), but the existing entry in LDAP will not be modified to reflect the new database. In this case, the administrator can:
  - Remove the existing database entry in LDAP and manually register the new database in LDAP.
  - Register the new database in LDAP using a different alias name.

## See Also

"CATALOG LDAP NODE" on page 169

"UNCATALOG LDAP DATABASE" on page 488

"UNCATALOG LDAP NODE" on page 490.

## CATALOG LDAP NODE

Catalogs a new node entry in LDAP (Lightweight Directory Access Protocol).

This command is available on Windows NT, Windows 98, and Windows 95 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──CATALOG LDAP──NODE──nodename──AS──nodealias─────────────────────────────►

►──┬──────────────────────────────────────┬───────────────────────────────►◄
   └─USER──username──┬──────────────────┬─┘
                     └─PASSWORD──password─┘
```

### Command Parameters

**NODE nodename**
: Specifies the LDAP node name of the DB2 server.

**AS nodealias**
: Specifies a new alias name for the LDAP node entry.

**USER username**
: Specifies the user's LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create the object in the LDAP directory. If the user's LDAP DN is not specified, the credentials of the current logon user will be used.

**PASSWORD password**
: Account password.

### Usage Notes

The CATALOG LDAP NODE command is used to specify a different alias name for the node that represents the DB2 server.

### See Also

"CATALOG LDAP DATABASE" on page 165

"UNCATALOG LDAP DATABASE" on page 488

"UNCATALOG LDAP NODE" on page 490.

## CATALOG LOCAL NODE

Creates a local alias for an instance that resides on the same machine. A local node should be cataloged when there is more than one instance on the same workstation to be accessed from the user's client. Interprocess Communications (IPC) is used to access the local node.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CATALOG──┬───────┬──LOCAL NODE──nodename──┬─────────────────────────┬──────────────►
             └─ADMIN─┘                        └─INSTANCE──instancename──┘

►──┬─────────────────────┬──┬──────────────────────────────┬──────────────────────────►
   └─SYSTEM──system-name─┘  └─OSTYPE──operating-system-type─┘

►──┬─────────────────────────┬──────────────────────────────────────────────────────►◄
   └─WITH──"comment-string"──┘
```

### Command Parameters

**ADMIN**
: Specifies that a local administration server node is to be cataloged.

**NODE nodename**
: A local alias for the node to be cataloged. This is an arbitrary name on the user's workstation, used to identify the node. It should be a meaningful name to make it easier to remember. The name must conform to database manager naming conventions (see "Appendix B. Naming Conventions" on page 525).

**INSTANCE instancename**
: Name of the local instance to be accessed.

**SYSTEM system-name**
: Specifies the DB2 system name that is used to identify the server machine.

**OSTYPE operating-system-type**
Specifies the operating system type of the server machine. Valid values are: OS2, AIX, WIN95, NT, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, and SGI.

**WITH ″comment-string″**
Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Examples

Workstation A has two server instances, `inst1` and `inst2`. To create databases at both instances from a single CLP session, issue the following sequence of commands (assume the **DB2INSTANCE** environment variable is set to `inst1`):

1. Create a local database at `inst1`:
   ```
   db2 create database mydb1
   ```
2. Catalog another server instance on this workstation:
   ```
   db2 catalog local node mynode2 instance inst2
   ```
3. Create a database at `mynode2`:
   ```
   db2 attach to mynode2
   db2 create database mydb2
   ```

## Usage Notes

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CATALOG NAMED PIPE NODE

Adds a named pipe node entry to the node directory. The named pipe is used to access the remote node.

This command is available on Windows NT only.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CATALOG──┬───────┬──NPIPE NODE──nodename──REMOTE──computername──────────────────►
             └─ADMIN─┘

►─INSTANCE──instancename──┬─────────────────────┬────────────────────────────────────►
                          └─SYSTEM──system-name─┘

►─┬─────────────────────────────────┬──┬───────────────────────────┬─────────────────►◄
  └─OSTYPE──operating-system-type───┘  └─WITH──"comment-string"────┘
```

### Command Parameters

**ADMIN**
Specifies that an NPIPE administration server node is to be cataloged.

**NODE nodename**
A local alias for the node to be cataloged. This is an arbitrary name on the user's workstation, used to identify the node. It should be a meaningful name to make it easier to remember. The name must conform to database manager naming conventions (see "Appendix B. Naming Conventions" on page 525).

**REMOTE computername**
The computer name of the node on which the target database resides. Maximum length is 15 characters.

**INSTANCE instancename**
Name of the server instance on which the target database resides.

Identical to the name of the remote named pipe, which is used to communicate with the remote node.

**SYSTEM system-name**
Specifies the DB2 system name that is used to identify the server machine.

**OSTYPE operating-system-type**
Specifies the operating system type of the server machine. Valid values are: OS2, AIX, WIN95, NT, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, and SGI.

**WITH ″comment-string″**
Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Examples

```
db2 catalog npipe node db2np1 remote nphost instance db2inst1
   with "A remote named pipe node."
```

## Usage Notes

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). On an OS/2 client or a Windows client, it stores and maintains the node directory in the instance subdirectory where the client is installed. On an AIX client, it creates the node directory in the DB2 installation directory.

List the contents of the local node directory using “LIST NODE DIRECTORY” on page 319.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CATALOG NETBIOS NODE

Adds a NetBIOS node entry to the node directory. The NetBIOS communications protocol is used to access the remote node.

This command is available on OS/2, Windows NT, and Windows 95 only.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None. Directory operations affect the local directory only.

### Command Syntax

```
►►──CATALOG──┬───────┬──NETBIOS NODE──nodename──REMOTE──server-nname──────────────►
             └─ADMIN─┘

►─ADAPTER──adapter-number──┬────────────────────────────────────┬─────────────────►
                           └─REMOTE_INSTANCE──instance-name─────┘

►─┬─────────────────────┬──┬─────────────────────────────────────┬────────────────►
  └─SYSTEM──system-name─┘  └─OSTYPE──operating-system-type───────┘

►─┬─────────────────────────┬──────────────────────────────────────────────────►◄
  └─WITH──"comment-string"──┘
```

### Command Parameters

**ADMIN**
: Specifies administration server nodes.

**NODE nodename**
: A local alias for the node to be cataloged. This is an arbitrary name on the user’s workstation, used to identify the node. It should be a meaningful name to make it easier to remember. The name must conform to database manager naming conventions (see “Appendix B. Naming Conventions” on page 525).

**REMOTE server-nname**
: The name of the remote workstation where the target database resides. This name must conform to the naming conventions for the

database manager. This is the workstation name (*nname*) found in the database manager configuration file of the server workstation.

**ADAPTER adapter-number**
Specifies the local, logical, outgoing LAN adapter number. The default value is zero.

**REMOTE_INSTANCE instance-name**
Specifies the real name of the instance to which an attachment is being made on the remote server machine.

**SYSTEM system-name**
Specifies a name that is used to identify the server machine.

**OSTYPE operating-system-type**
Specifies the operating system type of the server machine. Valid values are: OS2, AIX, WIN95, NT, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, and SGI.

**WITH ″comment-string″**
Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Examples

```
db2 catalog netbios node db2netb1 remote db2inst1 adapter 0
   with "A remote NetBIOS node"
```

## Usage Notes

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). On an OS/2 client or a Windows client, it stores and maintains the node directory in the instance subdirectory where the client is installed. On an AIX client, it creates the node directory in the DB2 installation directory.

List the contents of the local node directory using “LIST NODE DIRECTORY” on page 319.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP's directory cache, use “TERMINATE” on page 484. To refresh DB2's shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CATALOG ODBC DATA SOURCE

Catalogs a user or system ODBC data source.

A *data source*, in ODBC (Open Database Connectivity) terminology, is a user-defined name for a specific database or file system. That name is used to access the database or file system through ODBC APIs. On Windows NT and Windows 95, either user or system data sources can be cataloged. A user data source is only visible to the user who cataloged it, whereas a system data source is visible to and can be used by all other users.

This command is available on Windows NT, Windows 95, and Windows 3.1 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
                  ┌─USER───┐
►►──CATALOG───────┤        ├──ODBC DATA SOURCE──data-source-name──────────────────►◄
                  └─SYSTEM─┘
```

### Command Parameters

**USER** Catalog a user data source. This is the default if no keyword is specified.

**SYSTEM**
Catalog a system data source.

**ODBC DATA SOURCE data-source-name**
Specifies the name of the data source to be cataloged. Maximum length is 32 characters.

### See Also

"LIST ODBC DATA SOURCES" on page 325

"UNCATALOG ODBC DATA SOURCE" on page 493.

## CATALOG TCP/IP NODE

Adds a Transmission Control Protocol/Internet Protocol (TCP/IP) node entry to the node directory. The TCP/IP communications protocol is used to access the remote node.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None. Directory operations affect the local directory only.

### Command Syntax

```
►►──CATALOG──┬─────────┬──TCPIP NODE──nodename──REMOTE──hostname──────────────────►
             └─ADMIN───┘

   (1)
►────────SERVER──service-name──┬────────────────┬─────────────────────────────────►
                               └─SECURITY SOCKS─┘

►──┬─────────────────────────────────┬──┬─────────────────────────┬───────────────►
   └─REMOTE_INSTANCE──instance-name──┘  └─SYSTEM──system-name─────┘

►──┬───────────────────────────────────┬──┬─────────────────────────┬─────────────►◄
   └─OSTYPE──operating-system-type─────┘  └─WITH──"comment-string"──┘
```

**Notes:**

1. SERVER must not be specified for ADMIN nodes, but is mandatory for non-ADMIN nodes.

### Command Parameters

**ADMIN**
: Specifies that a TCP/IP administration server node is to be cataloged.

**NODE nodename**
: A local alias for the node to be cataloged. This is an arbitrary name on the user's workstation, used to identify the node. It should be a meaningful name to make it easier to remember. The name must conform to database manager naming conventions (see "Appendix B. Naming Conventions" on page 525).

**REMOTE hostname**
: The host name of the node where the target database resides. The host name is the name of the node that is known to the TCP/IP network. Maximum length is 255 characters.

**SERVER service-name**
: Specifies the service name or the port number of the server database manager instance.

The CATALOG TCPIP NODE command is run on a client.

- If a service name is specified, the *services* file on the client is used to map the service name to a port number. A service name is specified in the database manager configuration file, and the *services* file on the server is used to map this service name to a port number. The port number on the client and the server must match.

  **Note:** A port number, instead of a service name, can be specified in the database manager configuration file on the server, but this is not recommended.
- If a port number is specified, it must match the port number associated with the service name specified in the server's database manager configuration file. No service name needs to be specified in the local TCP/IP *services* file.

The value of *service-name* is used as a key to search the local *services* file for the associated port number. If a matching entry is not found, and *service-name* is numeric, the value is interpreted as the port number.

Maximum length is 14 characters. This parameter is case sensitive.

**Note:** This parameter must not be specified for ADMIN nodes. The value on ADMIN nodes is always 523.

**SECURITY SOCKS**
: Specifies that the node will be SOCKS-enabled.

The following environment variables are mandatory and *must* be set to enable SOCKS:

**SOCKS_NS**
: The Domain Name Server for resolving the host address of the SOCKS server. This should be an IP address.

**SOCKS_SERVER**
: The fully qualified host name or the IP address of the SOCKS

server. If the SOCKSified DB2 client is unable to resolve the fully qualified host name, it assumes that an IP address has been entered.

One of the following conditions should be true:

- The SOCKS server should be reachable via the domain name server
- It should be listed in the `hosts` file. The location of this file is described in the TCP/IP documentation.
- It should be in an IP address format.

If these environment variables are set after a **db2start** has been issued, it is necessary to issue a TERMINATE command.

**REMOTE_INSTANCE instance-name**
Specifies the name of the server instance to which an attachment is being made.

**SYSTEM system-name**
Specifies the DB2 system name that is used to identify the server machine.

**OSTYPE operating-system-type**
Specifies the operating system type of the server machine. Valid values are: OS2, AIX, WIN95, NT, HPUX, SUN, MVS, OS400, VM, VSE, SNI, SCO, and SGI.

**WITH ″comment-string″**
Describes the node entry in the node directory. Any comment that helps to describe the node can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Examples

```
db2 catalog tcpip node db2tcp1 remote tcphost server db2inst1
   with "A remote TCP/IP node"

db2 catalog tcpip node db2tcp2 remote 9.21.15.235 server db2inst2
   with "TCP/IP node using IP address"
```

## Usage Notes

The database manager creates the node directory when the first node is cataloged (that is, when the first CATALOG...NODE command is issued). On an OS/2 client or a Windows client, it stores and maintains the node directory in the instance subdirectory where the client is installed. On an AIX client, it creates the node directory in the DB2 installation directory.

List the contents of the local node directory using “LIST NODE DIRECTORY” on page 319.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database manager. To refresh the directory cache for another application, stop and then restart that application.

## CHANGE DATABASE COMMENT

Changes a database comment in the system database directory or the local database directory. New comment text can be substituted for text currently associated with a comment.

### Scope

This command only affects the node on which it is executed.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

None

### Command Syntax

```
►►──CHANGE──┬─DATABASE─┬──database-alias──COMMENT──┬─────────────────┬──────────────────►
            └─DB───────┘                            └─ON──┬─path──┬─┘
                                                          └─drive─┘

►─WITH──"comment-string"──────────────────────────────────────────────────────────────►◄
```

### Command Parameters

**DATABASE database-alias**
: Specifies the alias of the database whose comment is to be changed. To change the comment in the system database directory, specify the alias for the database. To change the comment in the local database directory, specify the path where the database resides (with the *path* parameter), and enter the name (not the alias) of the database.

**ON path/drive**
: On UNIX based systems, specifies the path on which the database resides, and changes the comment in the local database directory. If a path is not specified, the database comment for the entry in the system database directory is changed. On OS/2 or the Windows operating system, specifies the letter of the drive on which the database resides.

**WITH ″comment-string″**
: Describes the entry in the system database directory or the local

database directory. Any comment that helps to describe the cataloged database can be entered. The maximum length of a comment string is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

## Examples

The following example changes the text in the system database directory comment for the SAMPLE database from ″Test 2 - Holding″ to ″Test 2 - Add employee inf rows″:

```
db2 change database sample comment
   with "Test 2 - Add employee inf rows"
```

## Usage Notes

New comment text replaces existing text. To append information, enter the old comment text, followed by the new text.

Only the comment for an entry associated with the database alias is modified. Other entries with the same database name, but with different aliases, are not affected.

If the path is specified, the database alias must be cataloged in the local database directory. If the path is not specified, the database alias must be cataloged in the system database directory.

## See Also

"CREATE DATABASE" on page 187.

## CHANGE ISOLATION LEVEL

Changes the way that DB2 isolates data from other processes while a database is being accessed.

### Authorization

None

### Required Connection

None

### Command Syntax

```
                              ┌─CS─┐
►►──CHANGE──┬─SQLISL────┬──TO──┼─NC─┼──────────────────────────────────►◄
            └─ISOLATION─┘      ├─RR─┤
                               ├─RS─┤
                               └─UR─┘
```

### Command Parameters

**TO**

| | |
|---|---|
| **CS** | Specifies cursor stability as the isolation level. |
| **NC** | Specifies no commit as the isolation level. Not supported by DB2. |
| **RR** | Specifies repeatable read as the isolation level. |
| **RS** | Specifies read stability as the isolation level. |
| **UR** | Specifies uncommitted read as the isolation level. |

### Usage Notes

DB2 uses isolation levels to maintain data integrity in a database. The isolation level defines the degree to which an application process is isolated (shielded) from changes made by other concurrently executing application processes.

If a selected isolation level is not supported by a database, it is automatically escalated to a supported level at connect time.

Isolation level changes are not permitted while connected to a database with a type 1 connection (see “SET CLIENT” on page 465). Changes are permitted using a type 2 connection, but should be made with caution, because the changes will apply to every connection made from the same command line processor back-end process. The user assumes responsibility for remembering which isolation level applies to which connected database.

In the following example, a user is in DB2 interactive mode following creation of the SAMPLE database:

```
update command options using c off
catalog db sample as sample2

set client connect 2

connect to sample
connect to sample2

change isolation to cs
set connection sample
declare c1 cursor for select * from org
open c1
fetch c1 for 3 rows

change isolation to rr
fetch c1 for 2 rows
```

An SQL0514N error occurs because c1 is not in a prepared state for this isolation level.

```
change isolation to cs
set connection sample2
fetch c1 for 2 rows
```

An SQL0514N error occurs because c1 is not in a prepared state for this database.

```
declare c1 cursor for select division from org
```

A DB21029E error occurs because cursor c1 has already been declared and opened.

```
set connection sample
fetch c1 for 2 rows
```

This works because the original database (SAMPLE) was used with the original isolation level (CS).

For more information about isolation levels, see the *SQL Reference*.

## See Also

"QUERY CLIENT" on page 397.

## CREATE DATABASE

Initializes a new database with an optional user-defined collating sequence, creates the three initial table spaces, creates the system tables, and allocates the recovery log.

This command is not valid on a client.

### Scope

In a multi-node environment, this command affects all nodes that are listed in the `db2nodes.cfg` file.

The node from which this command is issued becomes the catalog node for the new database.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

Instance. To create a database at another (remote) node, it is necessary to first attach to that node. A database connection is temporarily established by this command during processing.

### Command Syntax

```
>>--CREATE--+-DATABASE-+--database-name--+------------------------------+-------><
            '-DB-------'                 +-AT NODE----------------------+
                                         '-| Create Database options |--'
```

**Create Database options:**

```
|--+-------------------+--+-------------------------+----------------------->
   '-ON--+-path--+-----'  '-ALIAS--database-alias---'
         '-drive-'

>--+------------------------------------------------+------------------------>
   '-USING CODESET--codeset--TERRITORY--territory---'
```

```
►─┬─────────────────────────────────────────┬─┬─────────────────────┬──────────────────────────►
  │                 ┌─SYSTEM────────┐       │ └─NUMSEGS──numsegs────┘
  └─COLLATE USING───┼─COMPATIBILITY─┼───────┘
                    └─IDENTITY──────┘

►─┬──────────────────────────────────┬─┬────────────────────────────────────────────┬──────────►
  └─DFT_EXTENT_SZ──dft_extentsize────┘ └─CATALOG TABLESPACE──┤ tblspace-defn ├──────┘

►─┬────────────────────────────────────────────┬────────────────────────────────────────────────►
  └─USER TABLESPACE──┤ tblspace-defn ├─────────┘

►─┬────────────────────────────────────────────────┬─┬──────────────────────────────┬──────────┤
  └─TEMPORARY TABLESPACE──┤ tblspace-defn ├────────┘ └─WITH──"comment-string"───────┘
```

**tblspace-defn:**

```
├──MANAGED BY───────────────────────────────────────────────────────────────────────────────────►

                          ┌─,──────────────────────┐
►──┬─SYSTEM USING──(──────▼─'container-string'─────┴──)──────────────────────────────────────┬──►
   │                        ┌─,───────────────────────────────────────────────────────┐       │
   └─DATABASE USING──(──────▼───┬─FILE───┬───'container-string' ──number-of-pages─────┴──)────┘
                                └─DEVICE─┘

►─┬──────────────────────────────┬─┬──────────────────────────────────┬─────────────────────────►
  └─EXTENTSIZE──number-of-pages──┘ └─PREFETCHSIZE──number-of-pages────┘

►─┬──────────────────────────────────────┬─┬──────────────────────────────────────────┬────────┤
  └─OVERHEAD──number-of-milliseconds─────┘ └─TRANSFERRATE──number-of-milliseconds─────┘
```

**Notes:**

1. The code set and territory values specified must be a valid combination. For a list of valid combinations, see one of the *Quick Beginnings* books.
2. For details on the **tblspace-defn** parameters, see the CREATE TABLESPACE statement in the *SQL Reference*. The table space definitions specified on CREATE DATABASE apply to all nodes on which the database is being created. They cannot be specified separately for each node. If the table space definitions are to be created differently on particular nodes, the CREATE TABLESPACE statement must be used.

When defining containers for table spaces, $N can be used. $N will be replaced by the node number when the container is actually created. This is required if the user wants to specify containers in a multiple logical node database.

## Command Parameters

**DATABASE database-name**
A name to be assigned to the new database. This must be a unique name that differentiates the database from any other database in either the local database directory or the system database directory. The name must conform to naming conventions for databases.

**AT NODE**
Specifies that the database is to be created only on the node that issues the command. This parameter is not intended for general use. For example, it should be used with "RESTORE DATABASE" on page 441 if the database partition at a node was damaged and must be re-created. Improper use of this parameter can cause inconsistencies in the system, so it should only be used with caution.

**Note:** If this parameter is used to recreate a database partition that was dropped (because it was damaged), the database at this node will be in the restore-pending state. After recreating the database partition, the database must immediately be restored on this node.

**ON path/drive**
On UNIX based systems, specifies the path on which to create the database. If a path is not specified, the database is created on the default database path specified in the database manager configuration file (*dftdbpath* parameter). Maximum length is 205 characters. On OS/2 or the Windows operating system, specifies the letter of the drive on which to create the database.

**Note:** For MPP systems, a database should not be created in an NFS-mounted directory. If a path is not specified, ensure that the *dftdbpath* database manager configuration parameter is not set to an NFS-mounted path (for example, on UNIX based systems, it should not specify the `$HOME` directory of the instance owner). The path specified for this command in an MPP system cannot be a relative path.

**ALIAS database-alias**
An alias for the database in the system database directory. If no alias is provided, the specified database name is used.

**USING CODESET codeset**
Specifies the code set to be used for data entered into this database.

**TERRITORY territory**
Specifies the territory to be used for data entered into this database.

**COLLATE USING**
Identifies the type of collating sequence to be used for the database. Once the database has been created, the collating sequence cannot be changed.

**COMPATIBILITY**
The DB2 Version 2 collating sequence. Some collation tables have been enhanced. This option specifies that the previous version of these tables is to be used.

**IDENTITY**
Identity collating sequence, in which strings are compared byte for byte.

**SYSTEM**
Collating sequence based on the current territory.

For information about how the database collating sequence is used, see the *SQL Reference*.

**NUMSEGS numsegs**
Specifies the number of segment directories that will be created and used to store DAT, IDX, LF, LB, and LBA files for any default SMS table spaces. This parameter does not affect DMS table spaces, any SMS table spaces with explicit creation characteristics (created when the database is created), or any SMS table spaces explicitly created after the database is created.

**DFT_EXTENT_SZ dft_extentsize**
Specifies the default extent size of table spaces in the database.

**CATALOG TABLESPACE tblspace-defn**
Specifies the definition of the table space which will hold the catalog tables, SYSCATSPACE. If not specified, SYSCATSPACE will be created as a System Managed Space (SMS) table space with *numsegs* number of directories as containers, and with an extent size of *dft_extentsize*. For example, the following containers would be created if *numsegs* were specified to be 5:

```
/u/smith/smith/NODE0000/SQL00001/SQLT0000.0
/u/smith/smith/NODE0000/SQL00001/SQLT0000.1
/u/smith/smith/NODE0000/SQL00001/SQLT0000.2
/u/smith/smith/NODE0000/SQL00001/SQLT0000.3
/u/smith/smith/NODE0000/SQL00001/SQLT0000.4
```

In an MPP system, the catalog table space is only created on the catalog node (the node on which the CREATE DATABASE command is issued).

**USER TABLESPACE tblspace-defn**
Specifies the definition of the initial user table space, USERSPACE1. If not specified, USERSPACE1 will be created as an SMS table space with *numsegs* number of directories as containers, and with an extent size of *dft_extentsize*. For example, the following containers would be created if *numsegs* were specified to be 5:

```
/u/smith/smith/NODE0000/SQL00001/SQLT0001.0
/u/smith/smith/NODE0000/SQL00001/SQLT0001.1
/u/smith/smith/NODE0000/SQL00001/SQLT0001.2
/u/smith/smith/NODE0000/SQL00001/SQLT0001.3
/u/smith/smith/NODE0000/SQL00001/SQLT0001.4
```

**TEMPORARY TABLESPACE tblspace-defn**
Specifies the definition of the initial temporary table space, TEMPSPACE1. If not specified, TEMPSPACE1 will be created as an SMS table space with *numsegs* number of directories as containers, and with an extent size of *dft_extentsize*. For example, the following containers would be created if *numsegs* were specified to be 5:

```
/u/smith/smith/NODE0000/SQL00001/SQLT0002.0
/u/smith/smith/NODE0000/SQL00001/SQLT0002.1
/u/smith/smith/NODE0000/SQL00001/SQLT0002.2
/u/smith/smith/NODE0000/SQL00001/SQLT0002.3
/u/smith/smith/NODE0000/SQL00001/SQLT0002.4
```

**WITH ″comment-string″**
Describes the database entry in the database directory. Any comment that helps to describe the database can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by single or double quotation marks.

## Usage Notes

CREATE DATABASE:

- Creates a database in the specified subdirectory. In an MPP system, creates the database on all nodes listed in `db2nodes.cfg`, and creates a `$DB2INSTANCE/NODE`*xxxx* directory under the specified subdirectory at each node. In a non-MPP system, creates a `$DB2INSTANCE/NODE0000` directory under the specified subdirectory.
- Creates the system catalog tables and recovery log.
- Catalogs the database in the following database directories:
  – server's local database directory on the path indicated by *path* or, if the path is not specified, the default database path defined in the database

manager system configuration file. A local database directory resides on each file system that contains a database.

– server's system database directory for the attached instance. The resulting directory entry will contain the database name and a database alias.

  If the command was issued from a remote client, the client's system database directory is also updated with the database name and an alias.

Creates a system or a local database directory if neither exists. If specified, the comment and code set values are placed in both directories.

- Stores the specified code set, territory, and collating sequence. A flag is set in the database configuration file if the collating sequence consists of unique weights, or if it is the identity sequence.
- Creates the schemata called SYSCAT, SYSFUN, SYSIBM, and SYSSTAT with SYSIBM as the owner. The server node on which this command is issued becomes the catalog node for the new database. Two nodegroups are created automatically: IBMDEFAULTGROUP and IBMCATGROUP. For more information, see the *SQL Reference*.
- Binds the previously defined database manager bind files to the database (these are listed in the utilities bind file list, `db2ubind.lst`). If one or more of these files do not bind successfully, CREATE DATABASE returns a warning in the SQLCA, and provides information about the binds that failed. If a bind fails, the user can take corrective action and manually bind the failing file. The database is created in any case. A schema called NULLID is implicitly created when performing the binds with CREATEIN privilege granted to PUBLIC.

  **Note:** The utilities bind file list contains two bind files that cannot be bound against down-level servers:
  - `db2ugtpi.bnd` cannot be bound against DB2 Version 2 servers.
  - `db2dropv.bnd` cannot be bound against DB2 Parallel Edition Version 1 servers.

  If `db2ubind.lst` is bound against a down-level server, warnings pertaining to these two files are returned, and can be disregarded.
- Creates SYSCATSPACE, TEMPSPACE1, and USERSPACE1 table spaces. The SYSCATSPACE table space is only created on the catalog node.
- Grants the following:
  - DBADM authority, and CONNECT, CREATETAB, BINDADD, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA privileges to the database creator
  - CONNECT, CREATETAB, BINDADD, and IMPLICIT_SCHEMA privileges to PUBLIC

– USE privilege on the USERSPACE1 table space to PUBLIC
– SELECT privilege on each system catalog to PUBLIC
– BIND and EXECUTE privilege to PUBLIC for each successfully bound utility.

With *dbadm* authority, one can grant these privileges to (and revoke them from) other users or PUBLIC. If another administrator with *sysadm* or *dbadm* authority over the database revokes these privileges, the database creator nevertheless retains them.

In an MPP environment, the database manager creates a subdirectory, `$DB2INSTANCE/NODE`*xxxx*, under the specified or default path on all nodes. The *xxxx* is the node number as defined in the `db2nodes.cfg` file (that is, node 0 becomes `NODE0000`). Subdirectories `SQL00001` through `SQL`*nnnnn* will reside on this path. This ensures that the database objects associated with different nodes are stored in different directories (even if the subdirectory `$DB2INSTANCE` under the specified or default path is shared by all nodes).

If LDAP (Lightweight Directory Access Protocol) support is enabled on the current machine, the database will be automatically registered in the LDAP directory. If a database object of the same name already exists in the LDAP directory, the database is still created on the local machine, but a warning message is returned, indicating that there is a naming conflict. In this case, the user can manually catalog an LDAP database entry by using “CATALOG LDAP DATABASE” on page 165.

CREATE DATABASE will fail if the application is already connected to a database.

Use CATALOG DATABASE to define different alias names for the new database.

## See Also

“BIND” on page 131

“CATALOG DATABASE” on page 153

“DROP DATABASE” on page 202.

## DEACTIVATE DATABASE

Stops the specified database.

### Scope

In an MPP system, this command deactivates the specified database on all nodes in the system. If one or more of these nodes encounters an error, a warning is returned. The database will be successfully deactivated on some nodes, but may remain activated on the nodes encountering the error.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

None

### Command Syntax

```
►►──DEACTIVATE──┬─DATABASE─┬──database-alias─────────────────────────────►
                └─DB───────┘

►─┬──────────────────────────────────────┬────────────────────────────►◄
  └─USER──username──┬──────────────────┬─┘
                    └─USING──password──┘
```

### Command Parameters

**DATABASE database-alias**
: Specifies the alias of the database to be stopped.

**USER username**
: Specifies the user stopping the database.

**USING password**
: Specifies the password for the user ID.

### Usage Notes

Databases initialized by ACTIVATE DATABASE can be shut down by DEACTIVATE DATABASE or by **db2stop**. If a database was initialized by ACTIVATE DATABASE, the last application disconnecting from the database will not shut down the database, and DEACTIVATE DATABASE must be used. (In this case, **db2stop** will also shut down the database.)

**Note:** The application issuing the DEACTIVATE DATABASE command cannot have an active database connection to any database.

## See Also

"ACTIVATE DATABASE" on page 116

"STOP DATABASE MANAGER" on page 480.

## DEREGISTER

Deregisters the DB2 server from the network directory server.

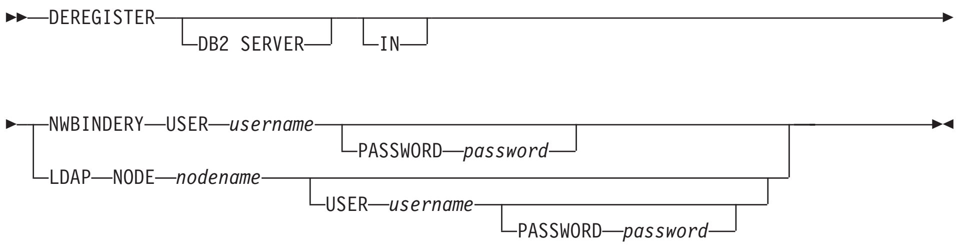
### Authorization

None

### Required Connection

None

### Command Syntax



### Command Parameters

**IN**
Specifies the network directory server from which to deregister the DB2 server. Valid values are: NWBINDERY for a NetWare bindery, and LDAP for an LDAP (Lightweight Directory Access Protocol) directory server.

**USER username**
For NWBINDERY, this is the user ID to log into the network server. The user ID must have SUPERVISOR or Workgroup Manager security equivalence. The user name must be provided when deregistering from the NetWare directory server, and is the user ID to log into the NETWARE server. For LDAP, this is the user's LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to delete the object from the LDAP directory. The user name is optional when deregistering in LDAP. If the user's LDAP DN is not specified, the credentials of the current logon user will be used.

**PASSWORD password**
Account password.

**NODE nodename**
The node name is the value that was specified when the DB2 server was registered in LDAP.

## Usage Notes

This command can only be issued for a remote machine when in the LDAP environment. When issued for a remote machine, the node name of the remote server must be specified.

The DB2 server is automatically deregistered when the instance is dropped.

## See Also

"REGISTER" on page 414

"UPDATE LDAP NODE" on page 505.

## DESCRIBE

This command:

- Displays the SQLDA information about a SELECT statement
- Displays columns of a table or a view
- Displays indexes of a table or a view

### Authorization

To display the SQLDA information about a SELECT statement, one of the privileges or authorities listed below for each table or view referenced in the SELECT statement is required.

To display the columns or indexes of a table or a view, one of the privileges or authorities listed below for the system catalogs `SYSCAT.COLUMNS` (DESCRIBE TABLE) and `SYSCAT.INDEXES` (DESCRIBE INDEXES FOR TABLE) is required:

- SELECT privilege
- CONTROL privilege
- *sysadm* or *dbadm* authority

### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

### Command Syntax

```
►►──DESCRIBE──┬─select-statement──────────────────────────────────┬──────────►◄
              └──┬─TABLE──table-name──────────┬──┬─────────────┬──┘
                 └─INDEXES FOR TABLE──table-name─┘  └─SHOW DETAIL─┘
```

### Command Parameters

**select-statement**
: Identifies the statement about which information is wanted. The SELECT statement is automatically prepared by CLP.

**TABLE table-name**
: Specifies the table or view to be described. The fully qualified name or alias in the form *schema.table-name* must be used. The *schema* is the user name under which the table or view was created.

  The DESCRIBE TABLE command lists the following information about each column:

  - Column name
  - Type schema

- Type name
- Length
- Scale
- Nulls (yes/no)

**INDEXES FOR TABLE table-name**
Specifies the table or view for which indexes need to be described. The fully qualified name or alias in the form *schema.table-name* must be used. The *schema* is the user name under which the table or view was created.

The DESCRIBE INDEXES FOR TABLE command lists the following information about each index of the table or view:

- Index schema
- Index name
- Unique rule
- Column count

**SHOW DETAIL**

For the DESCRIBE TABLE command, specifies that output include the following additional information:

- Whether a CHARACTER, VARCHAR or LONG VARCHAR column was defined as FOR BIT DATA
- Column number
- Partitioning key sequence
- Code page
- Default

For the DESCRIBE INDEXES FOR TABLE command, specifies that output include the following additional information:

- Column names

## Examples

### Describing a SELECT Statement

The following example shows how to describe a SELECT statement:

```
db2 "describe select * from staff"
```

```
SQLDA Information

sqldaid : SQLDA     sqldabc: 896  sqln: 20  sqld: 7

Column Information

sqltype               sqllen  sqlname.data                    sqlname.length
--------------------  ------  ------------------------------  --------------
500  SMALLINT              2  ID                                           2
449  VARCHAR               9  NAME                                         4
501  SMALLINT              2  DEPT                                         4
453  CHARACTER             5  JOB                                          3
501  SMALLINT              2  YEARS                                        5
485  DECIMAL            7, 2  SALARY                                       6
485  DECIMAL            7, 2  COMM                                         4
```

### Describing a Table

The following example shows how to describe a table:

```
   db2 describe table user1.department
```

```
Table: USER1.DEPARTMENT

Column             Type        Type
name               schema      name               Length   Scale    Nulls
------------------ ----------- ------------------ -------- -------- --------
AREA               SYSIBM      SMALLINT                  2        0 No
DEPT               SYSIBM      CHARACTER                 3        0 No
DEPTNAME           SYSIBM      CHARACTER                20        0 Yes
```

### Describing a Table Index

The following example shows how to describe a table index:

```
   db2 describe indexes for table user1.department
```

```
Table: USER1.DEPARTMENT

Index          Index               Unique         Number of
schema         name                rule           columns
-------------- ------------------  -------------- --------------
USER1          IDX1                U                           2
```

## DETACH

Removes the logical DBMS instance attachment, and terminates the physical communication connection if there are no other logical connections using this layer.

### Authorization

None

### Required Connection

None. Removes an existing instance attachment.

### Command Syntax

```
►►──DETACH─────────────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### See Also

"ATTACH" on page 121.

## DROP DATABASE

Deletes the database contents and all log files for the database, uncatalogs the database, and deletes the database subdirectory.

### Scope

By default, this command affects all nodes that are listed in the `db2nodes.cfg` file.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

Instance. An explicit attachment is not required. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command.

### Command Syntax

```
►►──DROP──┬─DATABASE─┬──database-alias──┬─────────┬──────────────────────────►◄
          └─DB───────┘                  └─AT NODE─┘
```

### Command Parameters

**DATABASE database-alias**
: Specifies the alias of the database to be dropped. The database must be cataloged in the system database directory.

**AT NODE**
: Specifies that the database is to be deleted only on the node that issued the DROP DATABASE command. This parameter is used by utilities supplied with DB2 Universal Database Enterprise - Extended Edition, and is not intended for general use. Improper use of this parameter can cause inconsistencies in the system, so it should only be used with caution.

### Examples

The following example deletes the database referenced by the database alias SAMPLE:

```
db2 drop database sample
```

## Usage Notes

DROP DATABASE deletes all user data and log files. If the log files are needed for a roll-forward recovery after a restore operation, the files should be saved prior to issuing this command.

The database must not be in use; all users must be disconnected from the database before the database can be dropped.

To be dropped, a database must be cataloged in the system database directory. Only the specified database alias is removed from the system database directory. If other aliases with the same database name exist, their entries remain. If the database being dropped is the last entry in the local database directory, the local database directory is deleted automatically.

If DROP DATABASE is issued from a remote client (or from a different instance on the same machine), the specified alias is removed from the client's system database directory. The corresponding database name is removed from the server's system database directory.

This command unlinks all files that are linked through any DATALINK columns. Since the unlink operation is performed asynchronously on the DB2 Data Links Manager, its effects may not be seen immediately on the DB2 Data Links Manager, and the unlinked files may not be immediately available for other operations. When the command is issued, all the DB2 Data Links Managers configured to that database must be available; otherwise, the drop database operation will fail.

## See Also

"CATALOG DATABASE" on page 153

"CREATE DATABASE" on page 187

"UNCATALOG DATABASE" on page 485.

## DROP NODE VERIFY

Verifies if a node exists in the nodegroups of any databases, and if an event monitor is defined on the node. This command should be used prior to dropping a node from an MPP system.

### Scope

This command only affects the node on which it is issued.

### Authorization

*sysadm*

### Command Syntax

```
►►──DROP NODE VERIFY──────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Usage Notes

If a message is returned, indicating that the node is not in use, use "STOP DATABASE MANAGER" on page 480 with DROP NODENUM to remove the entry for the node from the `db2nodes.cfg` file, which removes the node from the database system.

If a message is returned, indicating that the node is in use, the following actions should be taken:

1. If the node contains data, redistribute the data to remove it from the node using "REDISTRIBUTE NODEGROUP" on page 409. Use either the DROP NODE option on the REDISTRIBUTE NODEGROUP command, or the ALTER NODEGROUP statement to remove the node from any nodegroups for the database. This must be done for each database that contains the node in a nodegroup. For more information, see the *SQL Reference*.
2. Drop any event monitors that are defined on the node.
3. Rerun DROP NODE VERIFY to ensure that the database is no longer in use.

### See Also

"STOP DATABASE MANAGER" on page 480.

## ECHO

Permits the user to write character strings to standard output.

### Authorization

None

### Required Connection

None

### Command Syntax

```
>>--ECHO--+------------------+------------------------------------------><
          '-character-string-'
```

### Command Parameters

**character-string**
Any character string.

### Usage Notes

If an input file is used as standard input, or comments are to be printed without being interpreted by the command shell, the ECHO command will print character strings directly to standard output.

One line is printed each time that ECHO is issued.

The ECHO command is not affected by the verbose (-v) option (see "Command Line Processor Options" on page 97).

## EXPORT

Exports data from a database to one of several external file formats. The user specifies the data to be exported by supplying an SQL SELECT statement, or by providing hierarchical information for typed tables.

### Authorization

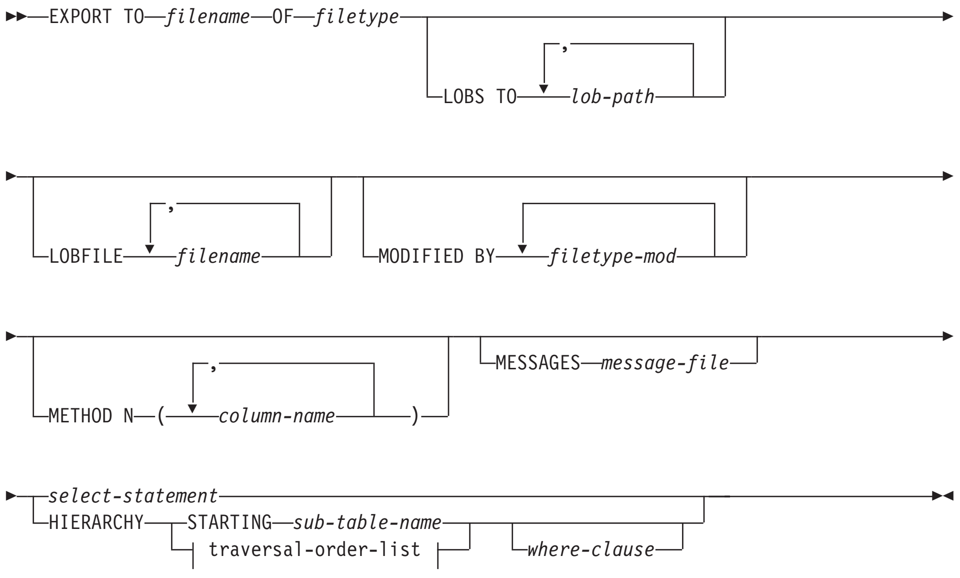
One of the following:
- *sysadm*
- *dbadm*

or CONTROL or SELECT privilege on each participating table or view.

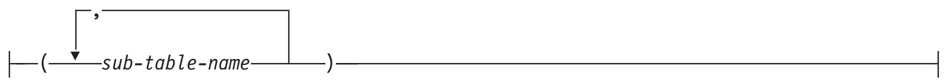
### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

### Command Syntax



**traversal-order-list:**



## Command Parameters

**HIERARCHY traversal-order-list**
Export a sub-hierarchy using the specified traverse order. All sub-tables must be listed in PRE-ORDER fashion. The first sub-table name is used as the target table name for the SELECT statement.

**HIERARCHY STARTING sub-table-name**
Using the default traverse order (OUTER order for ASC, DEL, or WSF files, or the order stored in PC/IXF data files), export a sub-hierarchy starting from *sub-table-name*.

**LOBFILE filename**
Specifies one or more base file names for the LOB files. When name space is exhausted for the first name, the second name is used, and so on.

When creating LOB files during an export operation, file names are constructed by appending the current base name from this list to the current path (from *lob-path*), and then appending a 3-digit sequence number. For example, if the current LOB path is the directory `/u/foo/lob/path`, and the current LOB file name is `bar`, the LOB files created will be `/u/foo/lob/path/bar.001`, `/u/foo/lob/path/bar.002`, and so on.

**LOBS TO lob-path**
Specifies one or more paths to directories in which the LOB files are to be stored. When file space is exhausted on the first path, the second path will be used, and so on.

**MESSAGES message-file**
Specifies the destination for warning and error messages that occur during an export operation. If the file already exists, the export utility appends the information. If *message-file* is omitted, the messages are written to standard output.

**METHOD N column-name**
Specifies one or more column names to be used in the output file. If this parameter is not specified, the column names in the table are used. This parameter is valid only for WSF and IXF files, but is not valid when exporting hierarchical data.

**MODIFIED BY filetype-mod**
Specifies additional options (see Table 5 on page 211).

**OF filetype**
Specifies the format of the data in the output file:

- DEL (delimited ASCII format), which is used by a variety of database manager and file manager programs.
- WSF (work sheet format), which is used by programs such as:

- Lotus 1-2-3
- Lotus Symphony

**Note:** When exporting BIGINT or DECIMAL data, only values that fall within the range of type DOUBLE can be exported accurately. Although values that do not fall within this range are also exported, importing or loading these values back may result in incorrect data, depending on the operating system.

- IXF (integrated exchange format, PC version), in which most of the table attributes, as well as any existing indexes, are saved in the IXF file, except when columns are specified in the SELECT statement. With this format, the table can be recreated, while with the other file formats, the table must already exist before data can be imported into it.

For more information about file formats, see the "Export/Import/Load Utility File Formats" appendix in the *Data Movement Utilities Guide and Reference.*

**select-statement**

Specifies the SELECT statement that will return the data to be exported. If the SELECT statement causes an error, a message is written to the message file (or to standard output). If the error code is one of SQL0012W, SQL0347W, SQL0360W, SQL0437W, or SQL1824W, the export operation continues; otherwise, it stops.

**TO filename**

Specifies the name of the file to which data is to be exported. If the complete path to the file is not specified, the export utility uses the current directory and the default drive as the destination.

If the name of a file that already exists is specified, the export utility overwrites the contents of the file; it does not append the information.

## Examples

The following example shows how to export information from the STAFF table in the SAMPLE database (to which the user must be connected) to `myfile.ixf`, with the output in IXF format. If the database connection is not through DB2 Connect, the index definitions (if any) will be stored in the output file; otherwise, only the data will be stored:

```
db2 export to myfile.ixf of ixf messages msgs.txt select * from staff
```

The following example shows how to export the information about employees in Department 20 from the STAFF table in the SAMPLE database (to which the user must be connected) to `awards.ixf`, with the output in IXF format:

```
db2 export to awards.ixf of ixf messages msgs.txt select * from staff
   where dept = 20
```

The following example shows how to export LOBs to an DEL file:

```
db2 export to myfile.del of del lobs to mylobs
   lobfile lobs1, lobs2 modified by lobsinfile
   select * from emp_photo
```

The following example shows how to export LOBs to a DEL file, specifying a second directory for files that may not fit into the first directory:

```
db2 export to myfile.del of del
   lobs to /db2exp1, /db2exp2 modified by lobsinfile
   select * from emp_photo
```

The following example shows how to export data to a DEL file, using a single quotation mark as the string delimiter, a semicolon as the column delimiter, and a comma as the decimal point. The same convention should be used when importing data back into the database:

```
db2 export to myfile.del of del
   modified by chardel'' coldel; decpt,
   select * from staff
```

## Usage Notes

Be sure to complete all table operations and release all locks before starting an export operation. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.

Table aliases can be used in the SELECT statement.

The messages placed in the message file include the information returned from the message retrieval service. Each message begins on a new line.

The export utility produces a warning message whenever a character column with a length greater than 254 is selected for export to DEL format files.

PC/IXF import should be used to move data between databases. If character data containing row separators is exported to a delimited ASCII (DEL) file and processed by a text transfer program (moving, for example, between OS/2 and AIX systems), fields containing the row separators will shrink or expand.

PC/IXF file format specifications permit migration of data between OS/2 (IBM Extended Services for OS/2, OS/2 Extended Edition and DB2 for OS/2) databases and DB2 for AIX databases via export, binary copying of files between OS/2 and AIX, and import. The file copying step is not necessary if the source and the target databases are both accessible from the same client.

DB2 Connect can be used to export tables from DRDA servers such as DB2 for OS/390, DB2 for VM and VSE, and DB2 for OS/400. Only PC/IXF export is supported.

The export utility will not create multiple-part PC/IXF files when invoked from an AIX system.

The export utility will store the NOT NULL WITH DEFAULT attribute of the table in an IXF file if the SELECT statement provided is in the form `SELECT * FROM tablename`.

When exporting typed tables, subselect statements can only be expressed by specifying the target table name and the WHERE clause. Fullselect and *select-statement* cannot be specified when exporting a hierarchy.

For file formats other than IXF, it is recommended that the traversal order list be specified, because it tells DB2 how to traverse the hierarchy, and what sub-tables to export. If this list is not specified, all tables in the hierarchy are exported, and the default order is the OUTER order. The alternative is to use the default order, which is the order given by the OUTER function.

**Note:** Use the same traverse order during an import operation. The load utility does not support loading hierarchies or sub-hierarchies.

**DB2 Data Links Manager Considerations**

To ensure that a consistent copy of the table and the corresponding files referenced by the DATALINK columns are copied for export, do the following:

1. Issue the command: QUIESCE TABLESPACES FOR TABLE tablename SHARE.

   This ensures that no update transactions are in progress when EXPORT is run.
2. Issue the EXPORT command.
3. Run the **dlfm_export** utility at each Data Links server. Input to the **dlfm_export** utility is the control file name, which is generated by the export utility. This produces a tar (or equivalent) archive of the files listed within the control file.
4. Issue the command: QUIESCE TABLESPACES FOR TABLE tablename RESET.

   This makes the table available for updates.

EXPORT is executed as an SQL application. The rows and columns satisfying the SELECT statement conditions are extracted from the database. For the DATALINK columns, the SELECT statement should not specify any scalar function.

Successful execution of EXPORT results in generation of the following files:

- An export data file as specified in the EXPORT command. A DATALINK column value in this file is in the format described on page 356. When the DATALINK column value is the SQL NULL value, handling is the same as that for other data types.
- Control files *server_name*, which are generated for each Data Links server (on the Windows NT operating system, a single control file, `ctrlfile.lst`, is used by all Data Links servers). These control files are placed in the directory <data-file path>/dlfm/YYYYMMDD/HHMMSS (on the Windows NT operating system, `ctrlfile.lst` is placed in the directory <data-file path>\dlfm\YYYYMMDD\HHMMSS). YYYYMMDD represents the date (year month day), and HHMMSS represents the time (hour minute second).

The **dlfm_export** utility is provided to export files from a Data Links server. This utility generates an archive file, which can be used to restore files in the target Data Links server.

Table 5. Valid File Type Modifiers (Export)

| **Modifier** | **Description** |
|---|---|
| | **All File Formats** |
| lobsinfile | *lob-path* specifies the path to the files containing LOB values. |
| | **DEL (Delimited ASCII) File Format** |
| chardel*x* | *x* is a single character string delimiter. The default value is a double quotation mark ("). The specified character is used in place of double quotation marks to enclose a character string.[a]<br><br>The single quotation mark (') can also be specified as a character string delimiter as follows:<br>`modified by chardel''` |
| coldel*x* | *x* is a single character column delimiter. The default value is a comma (,). The specified character is used in place of a comma to signal the end of a column.[a]<br><br>In the following example, `coldel;` causes the export utility to interpret any semicolon (;) it encounters as a column delimiter:<br>`db2 "export to temp of del modified by coldel;`<br>`   select * from staff where dept = 20"` |

Table 5. Valid File Type Modifiers (Export) (continued)

| **Modifier** | **Description** |
|---|---|
| datesiso | Date format. Causes all date data values to be exported in ISO format. |
| decplusblank | Plus sign character. Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign. |
| decpt*x* | *x* is a single character substitute for the period as a decimal point character. The default value is a period (.). The specified character is used in place of a period as a decimal point character.[a] |
| dldel*x* | *x* is a single character DATALINK delimiter. The default value is a semicolon (;). The specified character is used in place of a semicolon as the inter-field separator for a DATALINK value. It is needed because a DATALINK value may have more than one sub-value. [ab]<br>**Note:** *x* must not be the same character specified as the row, column, or character string delimiter. |
| nodoubledel | Suppresses recognition of double character delimiters. |
| **WSF File Format** | |
| 1 | Creates a WSF file that is compatible with Lotus 1-2-3 Release 1, or Lotus 1-2-3 Release 1a.[b] This is the default. |
| 2 | Creates a WSF file that is compatible with Lotus Symphony Release 1.0.[b] |
| 3 | Creates a WSF file that is compatible with Lotus 1-2-3 Version 2, or Lotus Symphony Release 1.1.[b] |
| 4 | Creates a WSF file containing DBCS characters. |

**Notes:**

1. The export utility does not issue a warning if an attempt is made to use unsupported file types with the MODIFIED BY option. If this is attempted, the export operation fails, and an error code is returned.
2. [a] “Delimiter Restrictions” on page 213 lists restrictions that apply to the characters that can be used as delimiter overrides.
3. [b] These files can also be directed to a specific product by specifying an L for Lotus 1-2-3, or an S for Symphony in the *filetype-mod* parameter string. Only one value or product designator may be specified.

## Delimiter Restrictions

It is the user's responsibility to ensure that the chosen delimiter character is not part of the data to be moved. If it is, unexpected errors may occur. The following restrictions apply to column, string, DATALINK, and decimal point delimiters when moving data:

- Delimiters are mutually exclusive.
- A delimiter cannot be binary zero, a line-feed character, a carriage-return, or a blank space.
- The default decimal point (.) cannot be a string delimiter.
- The following characters are specified differently by an ASCII-family code page and an EBCDIC-family code page:
  - The Shift-In (0x0F) and the Shift-Out (0x0E) character cannot be delimiters for an EBCDIC MBCS data file.
  - Delimiters for MBCS, EUC, or DBCS code pages cannot be greater than 0x40, except the default decimal point for EBCDIC MBCS data, which is 0x4b.
  - Default delimiters for data files in ASCII code pages or EBCDIC MBCS code pages are:

    ```
    " (0x22, double quotation mark; string delimiter)
    , (0x2c, comma; column delimiter)
    ```
  - Default delimiters for data files in EBCDIC SBCS code pages are:

    ```
    " (0x7F, double quotation mark; string delimiter)
    , (0x6B, comma; column delimiter)
    ```
  - The default decimal point for ASCII data files is 0x2e (period).
  - The default decimal point for EBCDIC data files is 0x4B (period).
  - If the code page of the server is different from the code page of the client, it is recommended that the hex representation of non-default delimiters be specified. For example,

    ```
    db2 load from ... modified by chardel0x0C coldelX1e ...
    ```

The following information about support for double character delimiter recognition in DEL files applies to the export, import, and load utilities:

- Character delimiters are permitted within the character-based fields of a DEL file. This applies to fields of type CHAR, VARCHAR, LONG VARCHAR, or CLOB (except when `lobsinfile` is specified). Any pair of character delimiters found between the enclosing character delimiters is imported or loaded into the database. For example,

  ```
  "What a ""nice"" day!"
  ```

  will be imported as:

  ```
  What a "nice" day!
  ```

In the case of export, the rule applies in reverse. For example,

```
I am 6" tall.
```

will be exported to a DEL file as:

```
"I am 6"" tall."
```

- In a DBCS environment, the pipe (|) character delimiter is not supported.

## See Also

"IMPORT" on page 273

"LOAD" on page 338.

## FORCE APPLICATION

Forces local or remote users or applications off the system to allow for maintenance on a server.

**Attention:** If an operation that cannot be interrupted (RESTORE DATABASE, for example) is forced, the operation must be successfully re-executed before the database becomes available.

### Scope

This command affects all nodes that are listed in the `$HOME/sqllib/db2nodes.cfg` file.

In a partitioned database environment, this command does not have to be issued from the coordinator node of the application being forced. It can be issued from any node (database partition server) in the partitioned database environment.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

Instance. To force users off a remote server, it is first necessary to attach to that server. If no attachment exists, this command is executed locally.

### Command Syntax

```
>>--FORCE APPLICATION--+-ALL-----------------------------+--+------------+----------><
                       |       .-,------------------.    |  '-MODE ASYNC-'
                       |       V                    |    |
                       '-(-------application-handle-+-)--'
```

### Command Parameters

**APPLICATION**

**ALL** All applications will be disconnected from the database.

**application-handle**
Specifies the agent to be terminated. List the values using "LIST APPLICATIONS" on page 295.

**MODE ASYNC**
The command does not wait for all specified users to be terminated before returning; it returns as soon as the function has been successfully issued or an error (such as invalid syntax) is discovered.

This is the only mode that is currently supported.

## Examples

The following example forces two users, with *application-handle* values of 41408 and 55458, to disconnect from the database:

```
   db2 force application ( 41408, 55458 )
```

## Usage Notes

**db2stop** cannot be executed during a force. The database manager remains active so that subsequent database manager operations can be handled without the need for **db2start**.

To preserve database integrity, only users who are idling or executing interruptible database operations can be terminated.

Users creating a database cannot be forced.

After a FORCE has been issued, the database will still accept requests to connect. Additional forces may be required to completely force all users off.

## See Also

"ATTACH" on page 121

"LIST APPLICATIONS" on page 295.

## GET ADMIN CONFIGURATION

Returns the values of individual entries in the database manager configuration file that are relevant to the DB2 Administration Server. The DB2 Administration Server is a special DB2 instance that enables remote administration of DB2 servers. The following database manager configuration parameters are displayed:

- AGENT_STACK_SZ
- AUTHENTICATION
- DIAGLEVEL
- DIAGPATH
- DISCOVER
- DISCOVER_COMM
- FILESERVER
- IPX_SOCKET
- NNAME
- OBJECTNAME
- QUERY_HEAP_SZ
- SVCENAME
- SYSADM_GROUP
- SYSCTRL_GROUP
- SYSMAINT_GROUP
- TPNAME
- TRUST_ALLCLNTS
- TRUST_CLNTAUTH

**Note:** The SVCENAME parameter, set by the installation program, cannot be modified by the user. The administration server service name is set to use the DB2 registered TCP/IP port (523).

For more information about these parameters, see “GET DATABASE MANAGER CONFIGURATION” on page 236.

### Scope

This command returns information on all nodes that share the same `$HOME/sqllib` directory, and can be issued from any of these nodes.

### Authorization

None

## GET ADMIN CONFIGURATION

### Required Connection

None or instance. An instance attachment is not required to perform local DBM configuration operations, but is required to perform remote DBM configuration operations. To display the database manager configuration for a remote instance, it is necessary to first attach to that instance.

### Command Syntax

```
►►──GET ADMIN───┬─CONFIGURATION─┬──────────────────────────────────────────────►◄
                ├─CONFIG────────┤
                └─CFG───────────┘
```

### Command Parameters

None

### Examples

The following is sample output from GET ADMIN CONFIGURATION:

```
            Admin Server Configuration

     Node type = Database Server with local and remote clients

 Database manager configuration release level            = 0x0900

 Diagnostic error capture level             (DIAGLEVEL) = 4
 Diagnostic data directory path              (DIAGPATH) =

 SYSADM group name                       (SYSADM_GROUP) =
 SYSCTRL group name                     (SYSCTRL_GROUP) =
 SYSMAINT group name                   (SYSMAINT_GROUP) =

 Database manager authentication       (AUTHENTICATION) = SERVER
 Cataloging allowed without authority   (CATALOG_NOAUTH) = NO
 Trust all clients                     (TRUST_ALLCLNTS) = YES
 Trusted client authentication         (TRUST_CLNTAUTH) = CLIENT

 Query heap size (4KB)                  (QUERY_HEAP_SZ) = 250
 TCP/IP Service name                         (SVCENAME) = 30676
 APPC Transaction program name                 (TPNAME) =
 IPX/SPX File server name                  (FILESERVER) =
 IPX/SPX DB2 server object name            (OBJECTNAME) =
 IPX/SPX Socket number                     (IPX_SOCKET) = 87A2

 Discovery mode                              (DISCOVER) = SEARCH
 Discovery communication protocols      (DISCOVER_COMM) = TCPIP
```

## Usage Notes

If an attachment to a remote instance (or a different local instance) exists, the admin configuration parameters for the attached server are returned; otherwise, the local admin configuration parameters are returned.

If an error occurs, the information returned is not valid. If the configuration file is invalid, an error message is returned. The user must install the database manager again to recover.

To set the configuration parameters to the default values shipped with the database manager, use "RESET ADMIN CONFIGURATION" on page 430.

For more information about these parameters, see the *Administration Guide*.

## See Also

"RESET ADMIN CONFIGURATION" on page 430

"UPDATE ADMIN CONFIGURATION" on page 494.

## GET AUTHORIZATIONS

Reports the authorities of the current user from values found in the database configuration file and the authorization system catalog view (SYSCAT.DBAUTH).

### Authorization

None

### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

### Command Syntax

```
►►──GET AUTHORIZATIONS──────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

The following is sample output from GET AUTHORIZATIONS:

```
 Administrative Authorizations for Current User

 Direct SYSADM authority             = NO
 Direct SYSCTRL authority            = NO
 Direct SYSMAINT authority           = NO
 Direct DBADM authority              = YES
 Direct CREATETAB authority          = YES
 Direct BINDADD authority            = YES
 Direct CONNECT authority            = YES
 Direct CREATE_NOT_FENC authority    = YES
 Direct IMPLICIT_SCHEMA authority    = YES

 Indirect SYSADM authority           = YES
 Indirect SYSCTRL authority          = NO
 Indirect SYSMAINT authority         = NO
 Indirect DBADM authority            = NO
 Indirect CREATETAB authority        = YES
 Indirect BINDADD authority          = YES
 Indirect CONNECT authority          = YES
 Indirect CREATE_NOT_FENC authority  = NO
 Indirect IMPLICIT_SCHEMA authority  = YES
```

## Usage Notes

Direct authorities are acquired by explicit commands that grant the authorities to a user ID. Indirect authorities are based on authorities acquired by the groups to which a user belongs.

**Note:** PUBLIC is a special group to which all users belong.

## GET CLI CONFIGURATION

Lists the contents of the `db2cli.ini` file. This command can list the entire file, or a specified section.

The `db2cli.ini` file is used as the DB2 call level interface (CLI) configuration file. It contains various keywords and values that can be used to modify the behavior of the DB2 CLI and the applications using it. The file is divided into sections, each section corresponding to a database alias name. For more information about this file and the CLI/ODBC configuration keywords, see the *CLI Guide and Reference*.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──GET CLI──┬─CONFIGURATION─┬──┬──────────────────────────┬──────────────►◄
             ├─CONFIG────────┤  └─FOR SECTION──section-name─┘
             └─CFG───────────┘
```

### Command Parameters

**FOR SECTION section-name**
: Name of the section whose keywords are to be listed. If not specified, all sections are listed.

### Examples

The following sample output represents the contents of a `db2cli.ini` file that has two sections:

```
Section: tstcli1x
-------------------------------------------------
  uid=userid
  pwd=*****
  autocommit=0
  TableType='TABLE','VIEW','SYSTEM TABLE'

Section: tstcli2x
-------------------------------------------------
  SchemaList='OWNER1','OWNER2',CURRENT SQLID
```

## Usage Notes

The section name specified on this command is not case sensitive. For example, if the section name in the db2cli.ini file (delimited by square brackets) is in lowercase, and the section name specified on the command is in uppercase, the correct section will be listed.

The value of the PWD (password) keyword is never listed; instead, five asterisks (*****) are listed.

When LDAP (Lightweight Directory Access Protocol) is enabled, the CLI configuration parameters can be set at the user level, in addition to the machine level. The CLI configuration at the user level is maintained in the LDAP directory. If the specified section exists at the user level, the CLI configuration for that section at the user level is returned; otherwise, the CLI configuration at the machine level is returned.

The CLI configuration at the user level is maintained in the LDAP directory and cached on the local machine. When reading the CLI configuration at the user level, DB2 always reads from the cache. The cache is refreshed when:

- The user updates the CLI configuration.
- The user explicitly forces a refresh of the CLI configuration using the REFRESH LDAP command.

## See Also

"REFRESH LDAP" on page 413

"UPDATE CLI CONFIGURATION" on page 497.

## GET CONNECTION STATE

Displays the connection state. Possible states are:

- Connectable and connected
- Connectable and unconnected
- Unconnectable and connected
- Implicitly connectable (if implicit connect is available).

For more information about these connection states, see the *SQL Reference*.

This command also returns information about the database connection mode (SHARE or EXCLUSIVE), and the alias and name of the database to which a connection exists (if one exists).

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──GET CONNECTION STATE──────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

The following is sample output from GET CONNECTION STATE:

```
   Database Connection State

 Connection state       = Connectable and Connected
 Connection mode        = SHARE
 Local database alias   = SAMPLE
 Database name          = SAMPLE
```

### Usage Notes

This command does not apply to type 2 connections (see “SET CLIENT” on page 465).

## GET DATABASE CONFIGURATION

Returns the values of individual entries in a specific database configuration file.

### Scope

This command returns information only for the node on which it is executed.

### Authorization

None

### Required Connection

Instance. An explicit attachment is not required. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command.

### Command Syntax

```
►►──GET──┬─DATABASE─┬──┬─CONFIGURATION─┬──FOR──database-alias──────────────────────►◄
         └─DB───────┘  ├─CONFIG────────┤
                       └─CFG───────────┘
```

### Command Parameters

**FOR database-alias**
: Specifies the alias of the database whose configuration is to be displayed.

### Examples

**Notes:**

1. Output on different platforms may show small variations reflecting platform-specific parameters.
2. Parameters with keywords enclosed by parentheses can be changed using “UPDATE DATABASE CONFIGURATION” on page 501.
3. Fields that do not contain keywords are maintained by the database manager and cannot be updated.

The following is sample output from GET DATABASE CONFIGURATION (issued on AIX):

```
          Database Configuration for Database sample

 Database configuration release level                    = 0x0900
 Database release level                                  = 0x0900
 Database territory                                      = C
 Database code page                                      = 819
 Database code set                                       = ISO8859-1
 Database country code                                   = 1

 Directory object name                    (DIR_OBJ_NAME) =
 Discovery support for this database       (DISCOVER_DB) = ENABLE

 Default query optimization class         (DFT_QUERYOPT) = 5
 Degree of parallelism                      (DFT_DEGREE) = 1
 Continue upon arithmetic exceptions   (DFT_SQLMATHWARN) = NO
 Number of frequent values retained     (NUM_FREQVALUES) = 10
 Number of quantiles retained            (NUM_QUANTILES) = 20

 Backup pending                                          = NO

 Database is consistent                                  = YES
 Rollforward pending                                     = NO
 Restore pending                                         = NO

 Multi-page file allocation enabled                      = NO

 Log retain for recovery status                          = NO
 User exit for logging status                            = NO

 Data Links Token Expiry Interval (sec)      (DL_EXPINT) = 60
 Data Links Number of Copies             (DL_NUM_COPIES) = 1
 Data Links Time after Drop (days)        (DL_TIME_DROP) = 1
 Data Links Token in Uppercase                (DL_UPPER) = NO
 Data Links Token Algorithm                   (DL_TOKEN) = MAC0

 Database heap (4KB)                            (DBHEAP) = 1200
 Catalog cache size (4KB)             (CATALOGCACHE_SZ) = 64
 Log buffer size (4KB)                        (LOGBUFSZ) = 8
 Utilities heap size (4KB)                (UTIL_HEAP_SZ) = 5000
 Buffer pool size (4KB)                       (BUFFPAGE) = 1000
 Extended storage segments size (4KB)    (ESTORE_SEG_SZ) = 16000
 Number of extended storage segments   (NUM_ESTORE_SEGS) = 0
 Max storage for lock list (4KB)              (LOCKLIST) = 100

 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 128

 Sort list heap (4KB)                         (SORTHEAP) = 256
 SQL statement heap (4KB)                     (STMTHEAP) = 2048
 Default application heap (4KB)             (APPLHEAPSZ) = 128
 Package cache size (4KB)                   (PCKCACHESZ) = (MAXAPPLS*8)
 Statistics heap size (4KB)               (STAT_HEAP_SZ) = 4384

 Interval for checking deadlock (ms)         (DLCHKTIME) = 10000
 Percent. of lock lists per application       (MAXLOCKS) = 10
 Lock timeout (sec)                        (LOCKTIMEOUT) = -1

 Changed pages threshold                (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 1
 Number of I/O servers                   (NUM_IOSERVERS) = 3
```

```
Index sort flag                              (INDEXSORT) = YES
Sequential detect flag                       (SEQDETECT) = YES
Default prefetch size (4KB)            (DFT_PREFETCH_SZ) = 32

Default number of containers                             = 1
Default tablespace extentsize (4KB)      (DFT_EXTENT_SZ) = 32

Max number of active applications             (MAXAPPLS) = 40
Average number of active applications        (AVG_APPLS) = 1
Max DB files open per application             (MAXFILOP) = 64

Log file size (4KB)                          (LOGFILSIZ) = 1000
Number of primary log files                 (LOGPRIMARY) = 3
Number of secondary log files                (LOGSECOND) = 2
Changed path to log files                   (NEWLOGPATH) =
Path to log files                                        = /home/smith/smith/
                                                           NODE0000/SQL00001/
                                                           SQLOGDIR/
First active log file                                    =

Group commit count                           (MINCOMMIT) = 1
Percent log file reclaimed before soft chckpt (SOFTMAX) = 100
Log retain for recovery enabled              (LOGRETAIN) = OFF
User exit for logging enabled                 (USEREXIT) = OFF

Auto restart enabled                       (AUTORESTART) = ON
Index re-creation time                        (INDEXREC) = SYSTEM (RESTART)
Default number of loadrec sessions     (DFT_LOADREC_SES) = 1
Number of database backups to retain    (NUM_DB_BACKUPS) = 12
Recovery history retention (days)      (REC_HIS_RETENTN) = 366

ADSM management class                   (ADSM_MGMTCLASS) =
ADSM node name                           (ADSM_NODENAME) =
ADSM owner                                  (ADSM_OWNER) =
ADSM password                            (ADSM_PASSWORD) =
```

These fields are identified below. Parameters whose name appears in lowercase are maintained by the database manager and cannot be updated. For more information about database configuration parameters, see the *Administration Guide*.

**ADSM_MGMTCLASS**
The ADSM management class specifies how the server should manage the backup versions or archive copies of the objects being backed up. The management class is assigned from the ADSM administrator. Once assigned, this parameter should be set to the management class name. When performing any ADSM backup, the database manager uses this parameter to pass the management class to ADSM.

**ADSM_NODENAME**
This parameter is used to override the default setting for the node

name associated with the ADSM product. The node name is needed when restoring a database that was backed up to ADSM from another node.

**ADSM_OWNER**
This parameter is used to override the default setting for the owner associated with the ADSM product. The owner name is needed when restoring a database that was backed up to ADSM from another node.

**ADSM_PASSWORD**
This parameter is used to override the default setting for the password associated with the ADSM product. The password is needed when restoring a database that was backed up to ADSM from another node.

**APP_CTL_HEAP_SZ**
This parameter determines the maximum size, in 4KB pages, for the application control heap. The heap is required to share information among agents working on behalf of the same application at a node in an MPP or an SMP system. If complex applications are being run, or the MPP configuration has a large number of nodes, the size of this heap should be increased.

**APPLHEAPSZ**
Specifies the size, in pages, of the application heap that is available for each individual agent.

**AUTORESTART**
Indicates whether the database manager can automatically issue RESTART DATABASE on a connect if, for example, the database connection was disrupted, or the database was not terminated normally during the previous session.

OFF specifies that it must be done manually.

ON specifies that the database manager does it automatically.

**AVG_APPLS**
Average number of active applications. Used by the SQL optimizer to help estimate how much buffer pool will be available for the chosen access plan at run time.

**backup_pending (Backup pending)**
NO specifies that the database is in a usable state.

YES specifies that an offline backup must be performed before the database can be used.

**BUFFPAGE**
Specifies the size, in pages, of the buffer pool. The buffer pool is used to store and manipulate data read in from the database. This

parameter is only used when a buffer pool's size has been explicitly set to -1 through either CREATE BUFFERPOOL, ALTER BUFFERPOOL, or "MIGRATE DATABASE" on page 370. The size of the buffer pool is normally controlled through SQL statements.

**CATALOGCACHE_SZ**
Controls the size, in pages, of the internal catalog cache (allocated from the *dbheap*), used by the SQL compiler to hold the packed descriptors for commonly referenced objects such as tables and constraints.

**CHNGPGS_THRESH**
Changed pages threshold. Used to specify the level (percentage) of changed pages at which the asynchronous page cleaners will be started, if they are not currently active.

**codepage (Database code page)**
Specifies the code page of the database.

**codeset (Database code set)**
Specifies the code set of the database.

**COPYPROTECT (OS/2 only)**
Enables the copy-protect attribute.

**country (Database country code)**
Specifies the country code of the database.

**database_consistent (Database is consistent)**
NO specifies that a transaction is pending, or some other task is pending on the database, and that the data is not consistent at this point.

YES specifies that all transactions have been committed or rolled back, and that the data is consistent.

**database_level (Database release level)**
Database release level. Specifies the release level of the database manager which can use the database.

**DBHEAP**
Specifies the size, in pages, of the database heap that is used to hold control information on all open cursors accessing the database. Both *logbufsz* and *catalogcache_sz* are allocated from the *dbheap*.

**DFT_DEGREE**
This parameter specifies the default value for the CURRENT DEGREE special register and the DEGREE bind option.

**DFT_EXTENT_SZ**
Default extent size of table spaces (in pages).

**DFT_LOADREC_SES**
Default number of load recovery sessions. Specifies the default number of sessions that will be used during the recovery of a table load. Applicable only if roll-forward recovery is enabled.

**DFT_PREFETCH_SZ**
Default prefetch size of table spaces (in pages).

**DFT_QUERYOPT**
The query optimization class is used to direct the optimizer to use different degrees of optimization when compiling SQL queries. This parameter provides additional flexibility by setting the default query optimization class used when neither the SET CURRENT QUERY OPTIMIZATION statement nor the QUERYOPT bind command are used.

**DIR_OBJ_NAME**
Object name in DCE name space. The object name representing a database manager instance (or a database) in the directory. The concatenation of this value and the *dir_path_name* value yields a global name that uniquely identifies the database manager instance or database in the name space governed by the directory services specified in the *dir_type* parameter.

**DISCOVER_DB**
This parameter can be set to `DISABLE` to prevent information about a database from being returned to a client when a discovery request is issued by the client against the server.

**DL_EXPINT**
Applies to DB2 Data Links Manager only. This parameter specifies the interval of time (in seconds) for which the file access token generated is valid. The number of seconds the token is valid begins from the time it is generated. The Data Links Filesystem Filter checks the validity of the token containing this expiry time.

**DL_NUM_COPIES**
Applies to DB2 Data Links Manager only. This parameter specifies the number of additional copies of a file to be made in the archive server (such as an ADSM server) when a file is linked to the database.

**DL_TIME_DROP**
Applies to DB2 Data Links Manager only. This parameter specifies the interval of time (in days) files would be retained on an archive server (such as an ADSM server) after a DROP TABLE, DROP DATABASE, or DROP TABLESPACE is issued.

**DL_TOKEN**
Applies to DB2 Data Links Manager only. This parameter specifies the algorithm used in the generation of DATALINK file access control tokens.

**DL_UPPER**
Applies to DB2 Data Links Manager only. This parameter indicates whether the file access control tokens use uppercase letters only, or whether the tokens can contain both uppercase and lowercase letters.

**DLCHKTIME**
Time interval (in milliseconds) for checking deadlock. Defines the frequency at which the database manager checks for deadlocks among all the applications connected to a database.

**ESTORE_SEG_SZ**
This parameter specifies the number of pages in each of the extended memory segments in the database. There are platform-dependent considerations when setting this configuration parameter.

**INDEXREC**
Specifies when invalid indexes will be recreated. The default setting is SYSTEM, which uses the value of the database manager configuration parameter *indexrec*.

The possible output values are:

- SYSTEM(ACCESS)
- SYSTEM(RESTART)
- ACCESS
- RESTART.

**INDEXSORT**
Index sort flag. Indicates whether sorting of index keys will occur during index creation.

**LOCKLIST**
Specifies the maximum storage, in pages, allocated to the lock list.

**LOCKTIMEOUT**
Specifies the number of seconds that an application will wait to obtain a lock.

**LOGBUFSZ**
Specifies the number of pages used to buffer log records prior to writing them to disk. Allocated from *dbheap*.

**LOGFILSIZ**
Specifies the amount of disk storage, in pages, allocated to log files used for data recovery. This parameter defines the size of each primary and secondary log file.

**loghead (First active log file)**
Log head identification. Specifies the name of the log file containing the head of the active log. The next log record that is written will start at the head of the active log.

**logpath (Path to log files)**
Location of log files. Contains the current path being used for logging purposes.

**LOGPRIMARY**
Specifies the number of primary log files that can be used for database recovery.

**LOGRETAIN**
Indicates whether the active log files are to be retained for use by roll-forward recovery, or for use by the data replication Capture program.

**log_retain_status (Log retain for recovery status)**
Indicates whether log files are being retained for use in roll-forward recovery.

**LOGSECOND**
Specifies the number of secondary log files that can be used for database recovery.

**MAXAPPLS**
Specifies the maximum number of application programs (both local and remote) that can connect to the database at one time.

**MAXFILOP**
Specifies the maximum number of database files that an application program can have open at one time.

**MAXLOCKS**
Specifies the maximum percentage of the lock list that any one application program can use.

**MINCOMMIT**
Specifies the number of SQL commits that can be grouped for a given database. Grouping SQL commits permits better control of the I/O and log activity when a commit is performed.

**MULTIPAGE_ALLOC**
Multi-page file allocation is used to improve insert performance. It applies to SMS table spaces only. If enabled, all SMS table spaces are affected: there is no selection possible for individual SMS table spaces.

**NEWLOGPATH**
Specifies an alternate path to the recovery log files for a database.

Since the *newlogpath* directory only accepts fully qualified directories, the absolute path must be specified.

**NUM_DB_BACKUPS**
This parameter specifies the number of database backups to retain for a database. After the specified number of backups is reached, old backups are marked as expired in the recovery history file. When a backup is marked as expired, the physical backups can be removed from where they are stored. The next database backup will prune the expired entries from the history file.

**NUM_ESTORE_SEGS**
This parameter specifies the number of extended storage memory segments available for use by the database.

**NUM_FREQVALUES**
Number of frequent values retained. Used to specify the number of ″most frequent values″ that will be collected when the WITH DISTRIBUTION option is specified in "RUNSTATS" on page 461.

**NUM_IOCLEANERS**
Specifies the number of asynchronous page cleaners for a database.

**NUM_IOSERVERS**
Specifies the number of I/O servers for a database. I/O servers are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities such as backup and restore.

**NUM_QUANTILES**
Number of quantiles for columns. Controls the number of quantiles that will be collected when the WITH DISTRIBUTION option is specified in "RUNSTATS" on page 461.

**numsegs (Default number of containers)**
Determines the number of containers that will be created within the default SMS table spaces.

**PCKCACHESZ**
Specifies the amount of memory to be used for caching packages and dynamic SQL statements.

The label (`calculated`) is displayed in the output for "GET DATABASE CONFIGURATION" on page 225 if:

- The internal value is -1
- MAXAPPLS*8 is less than 32. In this case, 32 is displayed with the label (`calculated`).

**REC_HIS_RETENTN**
Recovery history retention period. Used to specify the number of days that historical information on backups is to be retained.

**release (Database configuration release level)**
Specifies the release level of the database configuration file.

**restore_pending (Restore pending)**
This parameter indicates whether a RESTORE PENDING status exists in the database.

**rollfwd_pending (Rollforward pending)**
Indicates whether a roll-forward recovery procedure is required for the database.

The possible values are:

**NO** Neither the database nor any table space is in roll-forward pending state.

**DATABASE**
The database first needs to be rolled forward.

**TABLESPACES**
One or more table spaces in the database requires roll-forward recovery.

**SEQDETECT**
Indicates whether sequential detection for a database is to be enabled or disabled.

**SOFTMAX**
This parameter is used to specify the frequency at which soft checkpoints are taken, and to specify the number of logs that are to be recovered after a crash.

**SORTHEAP**
Specifies the number of private memory pages available for each sort in the application program.

**STAT_HEAP_SZ**
Statistics heap size (in pages). Specifies the maximum size of the heap used in creating and collecting all table statistics when distribution statistics are being gathered.

**STMTHEAP**
Specifies the heap size, in pages, that can be used for compiling SQL statements.

**territory (Database territory)**
Specifies the territory of the database.

**USEREXIT**
Indicates whether a user exit function for archiving or retrieving log files can be called the next time the database is opened.

OFF specifies that a user exit function cannot be called.

ON specifies that a user exit function can be called.

**user_exit_status (User exit for logging status)**
OFF specifies that the user exit function cannot be called to store archive log files.

ON specifies that the user exit function can be called to store archive log files.

**UTIL_HEAP_SZ**
Utility heap size. Specifies the maximum amount of shared memory that can be used simultaneously by the backup, restore, and load utilities.

## Usage Notes

If an error occurs, the information returned is not valid. If the configuration file is invalid, an error message is returned. The database must be restored from a backup version.

To set the database configuration parameters to the database manager defaults, use “RESET DATABASE CONFIGURATION” on page 433.

For more information about DB2 configuration parameters, see the *Administration Guide*.

## See Also

“RESET DATABASE CONFIGURATION” on page 433

“UPDATE DATABASE CONFIGURATION” on page 501.

## GET DATABASE MANAGER CONFIGURATION

Returns the values of individual entries in the database manager configuration file.

### Authorization

None

### Required Connection

None or instance. An instance attachment is not required to perform local DBM configuration operations, but is required to perform remote DBM configuration operations. To display the database manager configuration for a remote instance, it is necessary to first attach to that instance.

### Command Syntax

```
►►──GET──┬─DATABASE MANAGER─┬──┬─CONFIGURATION─┬──────────────────────────────►◄
         ├─DB MANAGER───────┤  ├─CONFIG────────┤
         └─DBM──────────────┘  └─CFG───────────┘
```

### Command Parameters

None

### Examples

**Note:** Both node type and platform determine which configuration parameters are listed.

The following is sample output from GET DATABASE MANAGER CONFIGURATION (issued on AIX):

```
          Database Manager Configuration

     Node type = Database Server with local clients

 Database manager configuration release level            = 0x0900

 CPU speed (millisec/instruction)             (CPUSPEED) = 4.000000e-05

 Max number of concurrently active databases     (NUMDB) = 8
 Data Links support                          (DATALINKS) = NO
 Federated Database System Support           (FEDERATED) = NO
 Transaction processor monitor name        (TP_MON_NAME) =

 Default charge-back account           (DFT_ACCOUNT_STR) =

 Java Development Kit 1.1 installation path (JDK11_PATH) =
 Diagnostic error capture level              (DIAGLEVEL) = 3
 Diagnostic data directory path               (DIAGPATH) =

 Default database monitor switches
   Buffer pool                         (DFT_MON_BUFPOOL) = OFF
   Lock                                   (DFT_MON_LOCK) = OFF
   Sort                                   (DFT_MON_SORT) = OFF
   Statement                              (DFT_MON_STMT) = OFF
   Table                                 (DFT_MON_TABLE) = OFF
   Unit of work                            (DFT_MON_UOW) = OFF

 SYSADM group name                        (SYSADM_GROUP) = BUILD
 SYSCTRL group name                      (SYSCTRL_GROUP) =
 SYSMAINT group name                    (SYSMAINT_GROUP) =

 Database manager authentication        (AUTHENTICATION) = SERVER
 Cataloging allowed without authority   (CATALOG_NOAUTH) = YES
 Trust all clients                      (TRUST_ALLCLNTS) = YES
 Trusted client authentication          (TRUST_CLNTAUTH) = CLIENT

 Default database path                       (DFTDBPATH) = /notnfs/mfarook

 Database monitor heap size (4KB)          (MON_HEAP_SZ) = 56
 UDF shared memory set size (4KB)           (UDF_MEM_SZ) = 256
 Java Virtual Machine heap size (4KB)     (JAVA_HEAP_SZ) = 512
 Audit buffer size (4KB)                  (AUDIT_BUF_SZ) = 0

 Backup buffer default size (4KB)            (BACKBUFSZ) = 1024
 Restore buffer default size (4KB)           (RESTBUFSZ) = 1024

 Sort heap threshold (4KB)                  (SHEAPTHRES) = 20000

 Directory cache support                     (DIR_CACHE) = YES

 Application support layer heap size (4KB)   (ASLHEAPSZ) = 15
 Max requester I/O block size (bytes)         (RQRIOBLK) = 32767
 Query heap size (4KB)                   (QUERY_HEAP_SZ) = 1000
 DRDA services heap size (4KB)            (DRDA_HEAP_SZ) = 128

 Priority of agents                           (AGENTPRI) = SYSTEM
 Max number of existing agents               (MAXAGENTS) = 200
 Agent pool size                        (NUM_POOLAGENTS) = 4 (calculated)
 Initial number of agents in pool       (NUM_INITAGENTS) = 0
 Initial number of fenced DARI process   (NUM_INITDARIS) = 0
 Max number of coordinating agents     (MAX_COORDAGENTS) = MAXAGENTS
 Max no. of concurrent coordinating agents  (MAXCAGENTS) = MAX_COORDAGENTS

 Keep DARI process                            (KEEPDARI) = YES
 Max number of DARI processes                  (MAXDARI) = MAX_COORDAGENTS
 Initialize DARI process with JVM         (INITDARI_JVM) = YES
 Index re-creation time                       (INDEXREC) = RESTART

 Transaction manager database name         (TM_DATABASE) = 1ST_CONN
 Transaction resync interval (sec)     (RESYNC_INTERVAL) = 180
```

```
 SPM name                                     (SPM_NAME) =
 SPM log size                          (SPM_LOG_FILE_SZ) = 256
 SPM resync agent limit                 (SPM_MAX_RESYNC) = 20
 SPM log path                             (SPM_LOG_PATH) =

 TCP/IP Service name                          (SVCENAME) =
 APPC Transaction program name                  (TPNAME) =
 IPX/SPX File server name                   (FILESERVER) =
 IPX/SPX DB2 server object name             (OBJECTNAME) =
 IPX/SPX Socket number                      (IPX_SOCKET) = 879E

 Discovery mode                               (DISCOVER) = SEARCH
 Discovery communication protocols       (DISCOVER_COMM) =
 Discover server instance                (DISCOVER_INST) = ENABLE

 Directory services type                      (DIR_TYPE) = NONE
 Directory path name                     (DIR_PATH_NAME) = /.:/subsys/database/
 Directory object name                    (DIR_OBJ_NAME) =
 Routing information object name        (ROUTE_OBJ_NAME) =
 Default client comm. protocols        (DFT_CLIENT_COMM) =

 Maximum query degree of parallelism   (MAX_QUERYDEGREE) = ANY
 Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

 No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = 512
 Number of FCM request blocks              (FCM_NUM_RQB) = 256
 Number of FCM connection entries      (FCM_NUM_CONNECT) = (FCM_NUM_RQB * 0.75)
 Number of FCM message anchors         (FCM_NUM_ANCHORS) = (FCM_NUM_RQB * 0.75)
```

These fields are identified as follows:

**AGENT_STACK_SZ (OS/2 only)**
The amount of memory allocated and committed by the operating system for each agent. This parameter specifies the number of pages for each agent stack on the server.

**AGENTPRI**
Execution priority assigned to database manager processes and threads on a particular machine.

**ASLHEAPSZ**
Size (in pages) of the memory shared between a local client application and a database manager agent.

**AUDIT_BUF_SZ**
Size (in pages) of the buffer used when auditing the database.

**AUTHENTICATION**
Determines how and where authentication of a user takes place. A value of CLIENT indicates that all authentication takes place at the

client. If the value is SERVER, the user ID and password are sent from the client to the server so that authentication can take place at the server.

**BACKBUFSZ**
Size (in pages) of the buffer used when backing up the database, if the buffer size is not specified when calling the backup utility.

**CATALOG_NOAUTH**
Specifies whether users are able to catalog and uncatalog databases and nodes, or DCS and ODBC directories, without SYSADM authority. The default value (0) for this parameter indicates that SYSADM authority is required. If this parameter is set to 1 (yes), SYSADM authority is not required.

**COMM_BANDWIDTH**
The value calculated for the communications bandwidth, in megabytes per second, is used by the SQL optimizer to estimate the cost of performing certain operations between the database partition servers of a partitioned database system.

**CONN_ELAPSE (MPP only)**
This parameter specifies the number of seconds within which a TCP/IP connection is to be established between two nodes. If the attempt completes within the time specified by this parameter, communications are established. If it fails, another attempt is made to establish communications. If the connection is attempted the number of times specified by the MAX_CONNRETRIES parameter and always times out, an error is returned.

**CPUSPEED**
CPU speed (in milliseconds per instruction) used by the SQL optimizer to estimate the cost of performing certain operations. The value of this parameter is set automatically when the database manager is installed, but can be modified to model a production environment on a test system, or to assess the impact of upgrading hardware.

**DATALINKS**
This parameter specifies whether Data Links support is enabled.

**DFT_ACCOUNT_STR**
Default accounting string.

**DFT_CLIENT_ADPT**
This parameter defines the default client adapter number for the NETBIOS protocol whose server nname is extracted from DCE Directory Services. This parameter can only be used with DCE.

**DFT_CLIENT_COMM**
Specifies the communication protocols that the client applications on a specific instance can use for remote connections. Used for configuring DCE only.

**DFT_MON_BUFPOOL**
Default value of the snapshot monitor's buffer pool switch.

**DFT_MON_LOCK**
Default value of the snapshot monitor's lock switch.

**DFT_MON_SORT**
Default value of the snapshot monitor's sort switch.

**DFT_MON_STMT**
Default value of the snapshot monitor's statement switch.

**DFT_MON_TABLE**
Default value of the snapshot monitor's table switch.

**DFT_MON_UOW**
Default value of the snapshot monitor's unit of work (UOW) switch.

**DFTDBPATH**
Default database path. If no path is specified when a database is created, the database is created on the path specified by this parameter.

**DIAGLEVEL**
Diagnostic error capture level determines the severity of diagnostic errors recorded in the error log file (`db2diag.log`).

**DIAGPATH**
The fully qualified path for DB2 diagnostic information.

**DIR_CACHE**
Directory cache support. If set to `YES`, database, node, and DCS directory files are cached in memory. This reduces connect costs by eliminating directory file I/O, and minimizing the directory searches required to retrieve directory information.

**DIR_OBJ_NAME**
Object name in DCE name space. The object name representing a database manager instance (or a database) in the directory. The concatenation of this value and the *dir_path_name* value yields a global name that uniquely identifies the database manager instance or database in the name space governed by the directory services specified in the *dir_type* parameter.

**DIR_PATH_NAME**
Directory path name in DCE name space. The unique name of the

database manager instance in the global name space is made up of this value and the value in the *dir_obj_name* parameter.

**DIR_TYPE**
Directory services type. Indicates whether the database manager instance uses the DCE global directory services.

**DISCOVER**
This parameter defines the type of discovery request supported on a client or server. Discovery requests can be issued from the client configuration assistant or from control center tools. Specify `SEARCH` to support search discovery, in which the DB2 client searches the network for DB2 databases. Specify `KNOWN` to support known discovery, in which the discovery request is issued against the administration server specified by the user. Specify `DISABLE` to disable the client or server from supporting any type of discovery request.

**DISCOVER_COMM**
This parameter defines the communications protocols that clients use to issue search discovery requests, and servers use to listen for search discovery requests. More than one protocol can be specified, separated by commas, or the parameter can be left blank. Supported protocols are TCPIP and NETBIOS.

**DISCOVER_INST**
This parameter enables or disables client discovery of an instance.

**DOS_RQRIOBLK**
DOS requester I/O block size. Applicable only on DOS clients, including DOS clients running under OS/2. This parameter controls the size of the I/O blocks that are allocated on the client and the server.

**DRDA_HEAP_SZ**
Specifies the size, in pages, of the DRDA heap. This heap is used by the DRDA AS and by DB2 Connect.

**FCM_NUM_ANCHORS**
This parameter specifies the number of FCM message anchors. Agents use the message anchors to send messages among themselves.

**FCM_NUM_BUFFERS**
This parameter specifies the number of 4KB buffers that are used for internal communications (messages) among the nodes in an instance.

**FCM_NUM_CONNECT**
This parameter specifies the number of FCM connection entries. Agents use connection entries to pass data among themselves.

**FCM_NUM_RQB**

This parameter specifies the number of FCM request blocks. Request blocks are the media through which information is passed between the FCM daemon and an agent.

**FEDERATED**

Federated database object support. When set to `YES`, the instance can use nicknames to access data managed by DB2 Family and other database managers.

**FILESERVER**

IPX/SPX file server name. Specifies the name of the Novell NetWare file server where the internetwork address of the database manager server instance is registered.

**Note:** The following characters are not valid: / \ : ; , * ?

**INDEXREC**

Specifies when invalid database indexes should be recreated. This parameter is used if the database configuration parameter *indexrec* is set to `SYSTEM`.

The possible output values are:

- ACCESS
- RESTART.

**INITDARI_JVM**

This parameter indicates whether each fenced DARI process will load the Java Virtual Machine (JVM) when starting. This parameter will reduce the initial startup time for fenced Java stored procedures, especially when used in conjunction with the *num_initdaris* parameter. This parameter could increase the initial load time for non-Java fenced stored procedures, because they do not need the JVM.

**INTRA_PARALLEL**

This parameter specifies whether the database manager can use intra-partition parallelism.

In a symmetric multiprocessor (SMP) environment, the default for this parameter is `YES`. In a non-SMP environment, the default for this parameter is `NO`. This parameter can be used on both partitioned and non-partitioned database systems. Some of the operations that can take advantage of parallel performance improvements when the value of this parameter is `YES` include database queries and index creation.

**IPX_SOCKET**

IPX/SPX socket number. Specifies a ″well-known″ socket number and represents the connection end point in a DB2 server's IPX/SPX internetwork address.

**JAVA_HEAP_SZ**
Determines the maximum size of the heap that is used by the Java interpreter. For non-partitioned database systems, one heap is allocated for the instance; for partitioned database systems, one heap is allocated for each database partition server.

**JDK11_PATH**
This parameter specifies the directory under which the Java Development Kit 1.1 is installed. The **CLASSPATH** and other environment variables used by the Java interpreter are computed from the value of this parameter.

**KEEPDARI**
Indicates whether to keep a DARI process after each DARI call. If `NO`, a new DARI process is created and terminated for each DARI invocation. If `YES`, a DARI process is reused for subsequent DARI calls, and is terminated only when the associated user application exits.

**MAX_CONNRETRIES (MPP only)**
If an attempt to establish communication between two nodes fails because the value specified by the CONN_ELAPSE parameter is reached (for example, the attempt to establish TCP/IP communication times out), MAX_CONNRETRIES specifies the number of connection retries that can be made to a node. If the value specified for this parameter is exceeded, an error is returned.

**MAX_COORDAGENTS**
This parameter determines the maximum number of coordinating agents that can exist at one time on a node.

**MAX_QUERYDEGREE**
This parameter specifies the maximum degree of parallelism used for any SQL statement executing on this instance of the database manager. An SQL statement will not use more than this number of parallel operations when the statement is executed. For a multi-node system, this parameter applies to the degree of parallelism within a single node.

**MAX_TIME_DIFF (MPP only)**
Each node has its own system clock. This parameter specifies the maximum time difference, in minutes, that is permitted among the nodes listed in the `db2nodes.cfg` file.

**MAXAGENTS**
Maximum number of database manager agents that can exist simultaneously on a node, regardless of which database is being used.

**MAXCAGENTS**
Maximum number of database manager agents that can be concurrently executing a database manager transaction. Cannot exceed the value of *maxagents*.

**MAXDARI**
Maximum number of DARI processes that can reside at the database server. Cannot exceed the value of *maxagents*.

**MAXTOTFILOP (OS/2 only)**
Maximum number of files open per application. Defines the total database and application file handles that can be used by a specific process connected to a database.

**MIN_PRIV_MEM (OS/2 only)**
Minimum committed private memory. Specifies the number of pages that the database server process will reserve as private virtual memory when a database manager instance is started (**db2start**).

**MON_HEAP_SZ**
Database system monitor heap size. Specifies the amount (in 4KB pages) of memory to allocate for database system monitor data.

**NNAME (OS/2 only)**
Name of the node or workstation. Database clients use *nname* to access database server workstations using NetBIOS. If the database server workstation changes the name specified in *nname*, all clients that access the database server workstation must catalog it again and specify the new name.

**nodetype (Node type)**
Indicates whether the node is configured as a database server with local and remote clients, a client, a database server with local clients, a partitioned database server with local and remote clients, or a Satellite database server with local clients.

**NOTIFYLEVEL (Windows NT only)**
This parameter is used to determine the severity of messages that are written to the notification files.

**NUM_INITAGENTS**
This parameter determines the initial number of agents that are created in the agent pool when the database manager is started.

**NUM_INITDARIS**
This parameter indicates the initial number of idle fenced DARI processes that are created in the DARI pool when the database manager is started. Setting this parameter will reduce the initial startup time for fenced stored procedures. This parameter is ignored if *keepdari* is not specified.

**NUM_POOLAGENTS**
This parameter specifies the size to which the agent pool is allowed to grow. The agent pool contains both idle agents (as in DB2/6000 Version 2), and MPP and SMP associated subagents. If more agents are created, they will be terminated and not return to the pool when they are finished executing.

If the value of this parameter is calculated at run time using other configuration parameters, the label `(calculated)` appears to the right of the value shown in the output for "GET DATABASE MANAGER CONFIGURATION" on page 236. If `-1 (calculated)` is shown in the output, the request was issued from a client, and the value was not available.

The obsolete database manager configuration parameter *max_idleagents* can still be updated through "UPDATE DATABASE MANAGER CONFIGURATION" on page 503, and is interpreted as an update to *num_poolagents*.

**NUMDB**
Maximum number of local databases that can be concurrently active (that is, have applications connected to them).

**OBJECTNAME**
This parameter represents the database manager server instance as an object on the NetWare file server, where the server's IPX/SPX address is stored and retrieved. The value must be entered in uppercase. The value must be unique on the NetWare file server, and it is recommended that it be unique across the IPX/SPX network.

**Note:** The following characters are not valid: / \ : ; , * ?

**PRIV_MEM_THRESH (OS/2 only)**
Private memory threshold. Sets a threshold below which a server will not release the memory associated with a client when that client's connection is terminated.

**QUERY_HEAP_SZ**
Maximum amount of memory (in pages) that can be allocated for the query heap. A query heap is used to store each query in the agent's private memory.

**release (Database manager configuration release level)**
Release level of the configuration file.

**RESTBUFSZ**
Size (in 4KB pages) of the buffer used when restoring the database, if the buffer size is not specified when calling the restore utility.

**RESYNC_INTERVAL**
Time interval (in seconds) after which a Transaction Manager (TM) or a Resource Manager (RM) retries the recovery of any outstanding indoubt transactions found in the TM or the RM. Applicable when transactions are running in a distributed unit of work (DUOW) environment.

**ROUTE_OBJ_NAME**
Routing information object name. Specifies the name of the default routing information object entry that will be used by all client applications attempting to access a DRDA server. Used for configuring DCE only.

**RQRIOBLK**
Client I/O block size. Specifies the size (in bytes) of the communication buffer between remote applications and their database agents on the database server.

**SHEAPTHRESH**
Limit on the total amount of memory (in pages) available for sorting across the entire instance.

**SPM_NAME**
This parameter identifies the name of the Sync Point Manager (SPM) instance to the database manager. The *spm_name* must be defined in the system database directory and, if remote, in the node directory.

**SPM_LOG_FILE_SZ**
This parameter identifies the Sync Point Manager (SPM) log file size in 4KB pages. The log file is contained in the `spmlog` sub-directory under `sqllib` and is created the first time SPM is started.

**SPM_LOG_PATH**
This parameter specifies the directory where the Sync Point Manager (SPM) logs are written. By default, the logs are written to the `sqllib` directory, which, in a high-volume transaction environment, can cause an I/O bottleneck. Use this parameter to have the SPM log files placed on a faster disk than the current `sqllib` directory. This allows for better concurrency among the SPM agents.

**SPM_MAX_RESYNC**
This parameter identifies the number of simultaneous agents that can perform resync operations.

**SS_LOGON (OS/2 only)**
By accepting the default for this parameter, a LOGON user ID and password are required before issuing a DB2START or DB2STOP.

**START_STOP_TIME (MPP only)**
This parameter specifies the time, in minutes, within which all nodes

must respond to "START DATABASE MANAGER" on page 475, "STOP DATABASE MANAGER" on page 480, or "ADD NODE" on page 119.

**SVCENAME**
The name used to update the database manager configuration file at the server. This value must be the same as the Connection Service name specified in the *services* file.

**SYSADM_GROUP**
Defines the group name with system administration (*sysadm*) authority for the database manager instance. This is the highest level of authority within the database manager, and controls all database objects.

**SYSCTRL_GROUP**
Defines the group name with system control (*sysctrl*) authority for the database manager instance. This level has privileges allowing operations affecting system resources, but not allowing direct access to data.

**SYSMAINT_GROUP**
Defines the group name with system maintenance (*sysmaint*) authority for the database manager instance. This level has authority allowing maintenance operations on all databases associated with an instance, but not allowing direct access to data.

**TM_DATABASE**
Name of the transaction manager (TM) database for each DB2 instance.

**TP_MON_NAME**
Name of the transaction processing (TP) monitor product being used.

**TPNAME**
Name of the remote transaction program that the database client must use when it issues an allocate request to the database manager instance using the APPC communication protocol.

**TRUST_ALLCLNTS**
This parameter and the TRUST_CLNTAUTH parameter are used to determine where users are validated for the database environment. By accepting the default for this parameter, all clients are treated as trusted clients. This means a level of security is available at the client, and that users can be validated at the client. Other options may be used to protect the server against certain clients based on their platform or database protocol.

**TRUST_CLNTAUTH**
This parameter and the TRUST_ALLCLNTS parameter are used to

determine where users are validated to the database environment. By accepting the default for this parameter, all users of trusted clients are validated at the client.

**UDF_MEM_SZ**
For a fenced user defined function (UDF), specifies the default allocation for memory to be shared between the database process and the UDF. For an unfenced process, specifies the size of the private memory set. In both cases, this memory is used to pass data to a UDF and back to a database.

## Usage Notes

If an attachment to a remote instance (or a different local instance) exists, the database manager configuration parameters for the attached server are returned; otherwise, the local database manager configuration parameters are returned.

If an error occurs, the information returned is invalid. If the configuration file is invalid, an error message is returned. The user must install the database manager again to recover.

To set the configuration parameters to the default values shipped with the database manager, use “RESET DATABASE MANAGER CONFIGURATION” on page 435.

For more information about these parameters, see the *Administration Guide*.

## See Also

“RESET DATABASE MANAGER CONFIGURATION” on page 435

“UPDATE DATABASE MANAGER CONFIGURATION” on page 503.

## GET DATABASE MANAGER MONITOR SWITCHES

Displays the status of the database system monitor switches. Monitor switches instruct the database system manager to collect database activity information. Each application using the database system monitor interface has its own set of monitor switches (see "GET MONITOR SWITCHES" on page 252). A database manager-level switch is on when any of the monitoring applications has turned it on. This command is used to determine if the database system monitor is currently collecting data for any monitoring application.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance or database:
- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To display the settings for a remote instance, or for a different local instance, it is necessary to first attach to that instance.

### Command Syntax

```
>>──GET──┬─DATABASE MANAGER─┬──MONITOR SWITCHES──────────────────────────────────><
         ├─DB MANAGER───────┤
         └─DBM──────────────┘
```

### Command Parameters

None

### Examples

The following is sample output from GET DATABASE MANAGER MONITOR SWITCHES:

```
            DBM System Monitor Information Collected

Buffer Pool Activity Information (BUFFERPOOL) = ON   06-11-1997 10:11:01.738377
Lock Information                       (LOCK) = OFF
Sorting Information                    (SORT) = ON   06-11-1997 10:11:01.738400
SQL Statement Information         (STATEMENT) = OFF
Table Activity Information            (TABLE) = OFF
Unit of Work Information                (UOW) = ON   06-11-1997 10:11:01.738353
```

## Usage Notes

The six recording switches (BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, and UOW) are off by default, but may be switched on using "UPDATE MONITOR SWITCHES" on page 508. If a particular switch is on, this command also displays the time stamp for when the switch was turned on.

For a summary of all database monitor data elements and monitoring groups, see the *System Monitor Guide and Reference*.

## See Also

"GET MONITOR SWITCHES" on page 252

"GET SNAPSHOT" on page 254

"RESET MONITOR" on page 437

"UPDATE MONITOR SWITCHES" on page 508.

## GET INSTANCE

Returns the value of the **DB2INSTANCE** environment variable.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──GET INSTANCE──────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

The following is sample output from GET INSTANCE:

```
 The current database manager instance is:  smith
```

## GET MONITOR SWITCHES

Displays the status of the database system monitor switches for the current session. Monitor switches instruct the database system manager to collect database activity information. Each application using the database system monitor interface has its own set of monitor switches. This command displays them. To display the database manager-level switches, use “GET DATABASE MANAGER MONITOR SWITCHES” on page 249.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To display the settings for a remote instance, or for a different local instance, it is necessary to first attach to that instance.

### Command Syntax

```
►►──GET MONITOR SWITCHES──────────────────────────────────────────────►◄
```

### Command Parameters

None

## Examples

The following is sample output from GET MONITOR SWITCHES:

```
            Monitor Recording Switches

Buffer Pool Activity Information  (BUFFERPOOL) = ON  02-20-1997 16:04:30.070073
Lock Information                        (LOCK) = OFF
Sorting Information                     (SORT) = OFF
SQL Statement Information          (STATEMENT) = ON  02-20-1997 16:04:30.070073
Table Activity Information             (TABLE) = OFF
Unit of Work Information                 (UOW) = ON  02-20-1997 16:04:30.070073
```

## Usage Notes

The six recording switches (BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, and UOW) are off by default, but may be switched on using "UPDATE MONITOR SWITCHES" on page 508. If a particular switch is on, this command also displays the time stamp for when the switch was turned on.

For a summary of all database monitor data elements and monitoring groups, see the *System Monitor Guide and Reference*.

## See Also

"GET DATABASE MANAGER MONITOR SWITCHES" on page 249

"GET SNAPSHOT" on page 254

"RESET MONITOR" on page 437

"UPDATE MONITOR SWITCHES" on page 508.

## GET SNAPSHOT

Collects database manager status information and returns it to a user-allocated data buffer. The information returned represents a *snapshot* of the database manager operational status at the time the command was issued.

### Scope

This command can be invoked from any node in the `db2nodes.cfg` file. It acts only on that node or partition.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To obtain a snapshot of a remote instance, it is necessary to first attach to that instance.

## Command Syntax

```
►►──GET SNAPSHOT FOR──────────────────────────────────────────────────────────►


►──┬──┬─DATABASE MANAGER─┬──────────────────────────────────────┬────────────►◄
   │  ├─DB MANAGER───────┤                                      │
   │  └─DBM──────────────┘                                      │
   ├─ALL──┬─────┬──DATABASES────────────────────────────────────┤
   │      └─DCS─┘                                               │
   ├─ALL──┬─────┬──APPLICATIONS─────────────────────────────────┤
   │      └─DCS─┘                                               │
   ├─ALL BUFFERPOOLS────────────────────────────────────────────┤
   ├──┬─────┬──APPLICATION────┬─APPLID──appl-id─────────┬───────┤
   │  └─DCS─┘                 └─AGENTID──appl-handle────┘       │
   ├─FCM FOR ALL NODES──────────────────────────────────────────┤
   ├─LOCKS FOR APPLICATION──┬─APPLID──appl-id─────────┬─────────┤
   │                        └─AGENTID──appl-handle────┘         │
   └──┬─ALL──────────────────────────────────┬──ON──database-alias──┘
      ├──┬─────┬──┬─DATABASE─┬───────────────┤
      │  └─DCS─┘  └─DB───────┘               │
      ├──┬─────┬──APPLICATIONS───────────────┤
      │  └─DCS─┘                             │
      ├─TABLES───────────────────────────────┤
      ├─TABLESPACES──────────────────────────┤
      ├─LOCKS────────────────────────────────┤
      ├─BUFFERPOOLS──────────────────────────┤
      └─DYNAMIC SQL──┬─────────────────┬─────┘
                     └─WRITE TO FILE───┘
```

**Note:** The monitor switches must be turned on to get some statistics (see "UPDATE MONITOR SWITCHES" on page 508).

## Command Parameters

**DATABASE MANAGER**
Provides statistics for the active database manager instance.

**ALL DATABASES**
Provides general statistics for all active databases on the current node.

**ALL APPLICATIONS**
Provides information about all active applications that are connected to a database on the current node.

**ALL BUFFERPOOLS**
Provides information about buffer pool activity for all active databases.

**APPLICATION APPLID appl-id**
Provides information only about the application whose ID is specified. To get a specific application ID, use "LIST APPLICATIONS" on page 295.

**APPLICATION AGENTID appl-handle**
Provides information only about the application whose application handle is specified. The application handle is a 32-bit number that uniquely identifies an application that is currently running. Use "LIST APPLICATIONS" on page 295 to get a specific application handle.

**FCM FOR ALL NODES**
Provides FCM statistics for all nodes.

**LOCKS FOR APPLICATION APPLID appl-id**
Provides information about all locks held by the specified application, identified by application ID.

**LOCKS FOR APPLICATION AGENTID appl-handle**
Provides information about all locks held by the specified application, identified by application handle.

**ALL ON database-alias**
Provides general statistics and information about all applications, tables, table spaces, buffer pools, and locks for a specified database.

**DATABASE ON database-alias**
Provides general statistics for a specified database.

**APPLICATIONS ON database-alias**
Provides information about all applications connected to a specified database.

**TABLES ON database-alias**
Provides information about tables in a specified database. This will include only those tables that have been accessed since the TABLE recording switch was turned on.

**TABLESPACES ON database-alias**
Provides information about table spaces for a specified database.

**LOCKS ON database-alias**
Provides information about every lock held by each application connected to a specified database.

**BUFFERPOOLS ON database-alias**
Provides information about buffer pool activity for the specified database.

**DYNAMIC SQL ON database-alias**
Returns a point-in-time picture of the contents of the SQL statement cache for the database.

**WRITE TO FILE**
Specifies that snapshot results are to be stored in a file at the server, as well as being passed back to the client. This command is valid only over a database connection. The snapshot data can then be queried through the table function SYSFUN.SQLCACHE_SNAPSHOT over the same connection on which the call was made. For more information, see the *System Monitor Guide and Reference*.

**DCS** Depending on which clause it is specified, this keyword requests statistics about:

- A specific DCS application currently running on the DB2 Connect Gateway
- All DCS applications
- All DCS applications currently connected to a specific DCS database
- A specific DCS database
- All DCS databases.

## Examples

In the following sample output listings, some of the information may not be available, depending on whether or not the appropriate database system monitor recording switch is turned on (see "UPDATE MONITOR SWITCHES" on page 508). If the information is unavailable, `Not Collected` appears in the output.

For more information about the fields displayed in the following output listings, see the *System Monitor Guide and Reference*.

The following is typical output resulting from a request for database manager information:

```
            Database Manager Snapshot

Node type                                       = Database Server with
                                                  local clients
Instance name                                   = smith
Number of nodes in DB2 instance                 = 0
Database manager status                         = Active

Product name                                    =
Product identification                          =
Service level                                   =

Sort heap allocated                             = 0
Post threshold sorts                            = 0
Piped sorts requested                           = 0
Piped sorts accepted                            = 0

Start Database Manager timestamp                = 02-25-1999 13:26:53.126518
```

```
Last reset timestamp                           =
Snapshot timestamp                             = 02-25-1999 13:45:42.257720

Remote connections to db manager               = 0
Remote connections executing in db manager     = 0
Local connections                              = 1
Local connections executing in db manager      = 0
Active local databases                         = 1

High water mark for agents registered          = 3
High water mark for agents waiting for a token = 0
Agents registered                              = 3
Agents waiting for a token                     = 0
Idle agents                                    = 1

Committed private Memory (Bytes)               = 3670016

Buffer Pool Activity Information  (BUFFERPOOL) = ON  02-25-1999 13:32:14
Lock Information                        (LOCK) = ON  02-25-1999 13:32:40
Sorting Information                     (SORT) = ON  02-25-1999 13:32:40
SQL Statement Information          (STATEMENT) = ON  02-25-1999 13:32:14
Table Activity Information             (TABLE) = ON  02-25-1999 13:32:40
Unit of Work Information                 (UOW) = ON  02-25-1999 13:32:14

Agents assigned from pool                      = 2
Agents created from empty pool                 = 3
Agents stolen from another application         = 0
High water mark for coordinating agents        = 3
Max agents overflow                            = 0
Hash joins after heap threshold exceeded       = 0

Total number of gateway connections            = 0
Current number of gateway connections          = 0
Gateway connections waiting for host reply     = 0
Gateway connections waiting for client reply   = 0
Gateway inactive connection pool agents        = 0
Gateway connection pool agents stolen          = 0
```

The following is typical output resulting from a request for database information:

```
              Database Snapshot

Database name                            = SAMPLE
Database path                            = /home/smith/smith/NODE0000/SQL00001/
Input database alias                     =
Database status                          = Active
Catalog node number                      = 0
Catalog network node name                =
Operating system running at database server= AIX
Location of the database                 = Local
First database connect timestamp         = 02-25-1999 13:31:33.886214
Last reset timestamp                     =
Last backup timestamp                    =
Snapshot timestamp                       = 02-25-1999 13:40:08.337902
```

```
High water mark for connections           = 1
Application connects                      = 1
Secondary connects total                  = 0
Applications connected currently          = 1
Appls. executing in db manager currently  = 0
Agents associated with applications       = 1
Maximum agents associated with applications= 1
Maximum coordinating agents               = 1

Locks held currently                      = 1
Lock waits                                = 0
Time database waited on locks (ms)        = 0
Lock list memory in use (Bytes)           = 432
Deadlocks detected                        = 0
Lock escalations                          = 0
Exclusive lock escalations                = 0
Agents currently waiting on locks         = 0
Lock Timeouts                             = 0

Total sort heap allocated                 = 0
Total sorts                               = 0
Total sort time (ms)                      = 0
Sort overflows                            = 0
Active sorts                              = 0

High water mark for database heap         = 316084

Buffer pool data logical reads            = 1
Buffer pool data physical reads           = 0
Asynchronous pool data page reads         = 0
Buffer pool data writes                   = 0
Asynchronous pool data page writes        = 0
Buffer pool index logical reads           = 0
Buffer pool index physical reads          = 0
Asynchronous pool index page reads        = 0
Buffer pool index writes                  = 0
Asynchronous pool index page writes       = 0
Total buffer pool read time (ms)          = 0
Total buffer pool write time (ms)         = 0
Total elapsed asynchronous read time      = 0
Total elapsed asynchronous write time     = 0
Asynchronous read requests                = 0
LSN Gap cleaner triggers                  = 0
Dirty page steal cleaner triggers         = 0
Dirty page threshold cleaner triggers     = 0
Time waited for prefetch (ms)             = 0
Direct reads                              = 0
Direct writes                             = 0
Direct read requests                      = 0
Direct write requests                     = 0
Direct reads elapsed time (ms)            = 0
Direct write elapsed time (ms)            = 0
Database files closed                     = 0
Data pages copied to extended storage     = 0
```

```
Index pages copied to extended storage     = 0
Data pages copied from extended storage    = 0
Index pages copied from extended storage   = 0

Commit statements attempted                = 2
Rollback statements attempted              = 0
Dynamic statements attempted               = 10
Static statements attempted                = 2
Failed statement operations                = 0
Select SQL statements executed             = 2
Update/Insert/Delete statements executed   = 0
DDL statements executed                    = 0

Internal automatic rebinds                 = 0
Internal rows deleted                      = 0
Internal rows inserted                     = 0
Internal rows updated                      = 0
Internal commits                           = 1
Internal rollbacks                         = 0
Internal rollbacks due to deadlock         = 0

Rows deleted                               = 0
Rows inserted                              = 0
Rows updated                               = 0
Rows selected                              = 16

Binds/precompiles attempted                = 0

Log space available to the database (Bytes)= 0
Log space used by the database (Bytes)     = 0
Maximum secondary log space used (Bytes)   = 0
Maximum total log space used (Bytes)       = 0
Secondary logs allocated currently         = 0
Log pages read                             = 0
Log pages written                          = 0
Appl id holding the oldest transaction     = 0

Package cache lookups                      = 2
Package cache inserts                      = 1
Package cache overflows                    = 0
Package cache high water mark (Bytes)      = 108757
Application section lookups                = 10
Application section inserts                = 1

Catalog cache lookups                      = 1
Catalog cache inserts                      = 1
Catalog cache overflows                    = 0
Catalog cache heap full                    = 0

Number of hash joins                       = 0
Number of hash loops                       = 0
Number of hash join overflows              = 0
Number of small hash join overflows        = 0
```

The following is typical output resulting from a request for DCS database information:

```
             DCS Database Snapshot

DCS database name                          = DCSDB
Host database name                         = GILROY
First database connect timestamp           = 02-25-1999 17:00:05.003421
Most recent elapsed time to connect        = 0.001200
Most recent elapsed connection duration    = 3.443780
Host response time (sec.ms)                = 0.000320
Last reset timestamp                       =
Number of SQL statements attempted         = 12
Commit statements attempted                = 6
Rollback statements attempted              = 2
Failed statement operations                = 4
Total number of gateway connections        = 0
Current number of gateway connections      = 1
Gateway conn. waiting for host reply       = 0
Gateway conn. waiting for client reply     = 1
Gateway communication errors to host       = 0
Timestamp of last communication error      = None
High water mark for gateway connections    = 1
Rows selected                              = 0
Outbound bytes sent                        = 0
Outbound bytes received                    = 0
```

The following is typical output resulting from a request for application information (by specifying either an application ID, an application handle, all applications, or all applications on a database):

```
            Application Snapshot

Application handle                         = 3
Application status                         = UOW Waiting
Status change time                         = 02-25-1999 13:33:41.446676
Application code page                      = 819
Application country code                   = 1
DUOW correlation token                     = *LOCAL.smith.990225183133
Application name                           = db2bp
Application ID                             = *LOCAL.smith.990225183133
Sequence number                            = 0001
Connection request start timestamp         = 02-25-1999 13:31:33.886214
Connect request completion timestamp       = 02-25-1999 13:31:34.434114
Application idle time                      = 6 minutes and 42 seconds
Authorization ID                           = SMITH
Client login ID                            = smith
Configuration NNAME of client              =
Client database manager product ID         = SQL06000
Process ID of client application           = 27918
Platform of client application             = AIX
Communication protocol of client           = Local Client

Outbound communication address             =
Outbound communication protocol            = APPC
```

```
Inbound communication address             =

Database name                             = SAMPLE
Database path                             = /home/smith/smith/
                                            NODE0000/SQL00001/
Client database alias                     = sample
Input database alias                      =
Last reset timestamp                      =
Snapshot timestamp                        = 02-25-1999 13:40:23.773540
The highest authority level granted       =
        Direct DBADM authority
        Direct CREATETAB authority
        Direct BINDADD authority
        Direct CONNECT authority
        Direct CREATE_NOT_FENC authority
        Direct IMPLICIT_SCHEMA authority
        Indirect SYSADM authority
        Indirect CREATETAB authority
        Indirect BINDADD authority
        Indirect CONNECT authority
        Indirect IMPLICIT_SCHEMA authority
Coordinating node number                  = 0
Current node number                       = 0
Coordinator agent process or thread ID    = 26160
Agents stolen                             = 0
Agents waiting on locks                   = 0
Maximum associated agents                 = 1
Priority at which application agents work = 0
Priority type                             = Dynamic

Locks held by application                 = 1
Lock waits since connect                  = 0
Time application waited on locks (ms)     = 0
Deadlocks detected                        = 0
Lock escalations                          = 0
Exclusive lock escalations                = 0
Number of Lock Timeouts since connected   = 0
Total time UOW waited on locks (ms)       = 0

Total sorts                               = 0
Total sort time (ms)                      = 0
Total sort overflows                      = 0

Data pages copied to extended storage     = 0
Index pages copied to extended storage    = 0
Data pages copied from extended storage   = 0
Index pages copied from extended storage  = 0
Buffer pool data logical reads            = 1
Buffer pool data physical reads           = 0
Buffer pool data writes                   = 0
Buffer pool index logical reads           = 0
Buffer pool index physical reads          = 0
Buffer pool index writes                  = 0
Total buffer pool read time (ms)          = 0
Total buffer pool write time (ms)         = 0
```

```
Time waited for prefetch (ms)              = 0
Direct reads                               = 0
Direct writes                              = 0
Direct read requests                       = 0
Direct write requests                      = 0
Direct reads elapsed time (ms)             = 0
Direct write elapsed time (ms)             = 0

Number of SQL requests since last commit   = 5
Commit statements                          = 2
Rollback statements                        = 0
Dynamic SQL statements attempted           = 10
Static SQL statements attempted            = 2
Failed statement operations                = 0
Select SQL statements executed             = 2
Update/Insert/Delete statements executed   = 0
DDL statements executed                    = 0
Internal automatic rebinds                 = 0
Internal rows deleted                      = 0
Internal rows inserted                     = 0
Internal rows updated                      = 0
Internal commits                           = 1
Internal rollbacks                         = 0
Internal rollbacks due to deadlock         = 0
Binds/precompiles attempted                = 0
Rows deleted                               = 0
Rows inserted                              = 0
Rows updated                               = 0
Rows selected                              = 16
Rows read                                  = 25
Rows written                               = 0

UOW log space used (Bytes)                 = 0
Previous UOW completion timestamp          = 02-25-1999 13:31:34.434114
Elapsed time of last completed uow (sec.ms)= 0.919533380
UOW start timestamp                        = 02-25-1999 13:33:41.392167
UOW stop timestamp                         =
UOW completion status                      =
Open remote cursors                        = 0
Open remote cursors with blocking          = 0
Rejected Block Remote Cursor requests      = 0
Accepted Block Remote Cursor requests      = 2
Open local cursors                         = 0
Open local cursors with blocking           = 0

Total User CPU Time used by agent (s)      = 0.100000
Total System CPU Time used by agent (s)    = 0.020000
Package cache lookups                      = 2
Package cache inserts                      = 1
Application section lookups                = 10
Application section inserts                = 1
Catalog cache lookups                      = 1
Catalog cache inserts                      = 1
Catalog cache overflows                    = 0
Catalog cache heap full                    = 0
```

```
Most recent operation                     = Select
Most recent operation start timestamp     = 02-25-1999 13:33:41.394260
Most recent operation stop timestamp      = 02-25-1999 13:33:41.446740
Agents associated with the application    = 1

Number of hash joins                      = 0
Number of hash loops                      = 0
Number of hash join overflows             = 0
Number of small hash join overflows       = 0

Statement type                            = Dynamic SQL Statement
Statement                                 = Select
Section number                            = 201
Application creator                       = NULLID
Package name                              = SQLC28A4
Cursor name                               = SQLCUR201
Statement node number                     = 0
Statement start timestamp                 = 02-25-1999 13:33:41.394260
Statement stop timestamp                  = 02-25-1999 13:33:41.446740
Elapsed time of last completed stmt(sec.ms)= 0.000000
Total user CPU time                       = 0.000000
Total system CPU time                     = 0.000000
SQL compiler cost estimate in timerons    = 30
SQL compiler cardinality estimate         = 47
Degree of parallelism requested           = 1
Number of agents working on statement     = 1
Number of subagents created for statement  = 1
Statement sorts                           = 0
Total sort time                           = 0
Sort overflows                            = 0
Rows read                                 = 8
Rows written                              = 0
Rows deleted                              = 0
Rows updated                              = 0
Rows inserted                             = 0
Rows fetched                              = 0
Number of subsections                     = 0
Dynamic SQL statement text:
select * from org
```

The following is typical output resulting from a request for DCS application information (by specifying either a DCS application ID, a DCS application handle, all DCS applications, or all DCS applications on a database):

```
           DCS Application Snapshot
Client application ID                     = 09151251.04D6.980521202839
  Sequence number                         = 0001
  Authorization ID                        = NEWTON
  Application name                        = db2bp
  Application handle                      = 0
  Application status                      = waiting for request
  Status change time                      = 05-21-1998 16:35:27.670354
  Client DB alias                         = MVSDB
  Client node                             = antman
  Client release level                    = SQL05020
  Client platform                         = AIX
  Client protocol                         = TCP/IP
  Client codepage                         = 819
  Process ID of client application        = 35754
  Client login ID                         = user1
  Host application ID                     = G9151251.G4D7.980521202840
  Sequence number                         = 0000
  Host DB name                            = GILROY
  Host release level                      = DSN05011
  Host CCSID                              = 500
Outbound communication address            = 9.21.21.92 5021
Outbound communication protocol           = TCP/IP
Inbound communication address             = 9.31.12.34 334
First database connect timestamp          = 05-21-1998 16:28:39.517919
Time spent on gateway processing          = 0.334215
Last reset timestamp                      =
Rows selected                             = 0
Number of SQL statements attempted        = 2
Failed statement operations               = 0
Commit statements                         = 1
Rollback statements                       = 0
Inbound bytes received                    = 392
Outbound bytes sent                       = 136
Outbound bytes received                   = 178
Inbound bytes sent                        = 190
Number of open cursors                    = 0
Application idle time                     = 53 seconds
UOW completion status                     = Committed - Commit Statement
Previous UOW completion timestamp         =
UOW start timestamp                       = 05-21-1998 16:35:27.252375
UOW stop timestamp                        = 05-21-1998 16:35:27.670290
Inbound bytes received for UOW            = 180
Outbound bytes sent for UOW               = 136
Outbound bytes received for UOW           = 178
Inbound bytes sent for UOW                = 190
Most recent operation                     = Static Commit
Most recent operation start timestamp     = 05-21-1998 16:35:27.284183
Most recent operation stop timestamp      = 05-21-1998 16:35:27.670290
Statement                                 = Static Commit
Section number                            = 0
Application creator                       = NULLID
Package name                              = SQLC28A0
SQL compiler cost estimate in timerons    = 0
SQL compiler cardinality estimate         = 0
Statement start timestamp                 = 05-21-1998 16:35:27.284183
Statement stop timestamp                  = 05-21-1998 16:35:27.670290
Rows fetched                              = 0
Time spent on gateway processing          = 0.333740
Inbound bytes received for statement      = 0
Outbound bytes sent for statement         = 10
Outbound bytes received for statement     = 54
Inbound bytes sent for statement          = 0
```

The following is typical output resulting from a request for buffer pool information:

```
             Bufferpool Snapshot
Bufferpool name                           = IBMDEFAULTBP
Database name                             = SAMPLE
Database path                             = /home/user1/user1/...
                                            NODE0000/SQL00011/
Input database alias                      = SAMPLE
Buffer pool data logical reads            = 32
Buffer pool data physical reads           = 13
Buffer pool data writes                   = 0
Buffer pool index logical reads           = 55
Buffer pool index physical reads          = 23
Total buffer pool read time (ms)          = 364
Total buffer pool write time (ms)         = 0
Database files closed                     = 0
Asynchronous pool data page reads         = 0
Asynchronous pool data page writes        = 0
Buffer pool index writes                  = 0
Asynchronous pool index page reads        = 0
Asynchronous pool index page writes       = 0
Total elapsed asynchronous read time      = 0
Total elapsed asynchronous write time     = 0
Asynchronous read requests                = 0
Direct reads                              = 34
Direct writes                             = 0
Direct read requests                      = 4
Direct write requests                     = 0
Direct reads elapsed time (ms)            = 1
Direct write elapsed time (ms)            = 0
Data pages copied to extended storage     = 0
Index pages copied to extended storage    = 0
Data pages copied from extended storage   = 0
Index pages copied from extended storage  = 0
```

The following is typical output resulting from a request for table information:

```
            Table Snapshot
First database connect timestamp     = 04-04-1997 14:29:55.197659
Last reset timestamp                 =
Snapshot timestamp                   = 04-04-1997 14:32:14.151875
Database name                        = SAMPLE
Database path                        = /home/user1/user1/NODE0000/SQL00011/
Input database alias                 = SAMPLE
Number of accessed tables            = 6
Table Schema    Table Name           Table Type   Rows Written  Rows Read  Overflows
------------... -------------------- ----------...------------ ---------- ----------
USER1           STAFF                User                    0         35          0
USER1           ORG                  User                    0          8          0
SYSIBM          SYSTABLES            Catalog                 0          2          0
SYSIBM          SYSTABLESPACES       Catalog                 0          3          0
SYSIBM          SYSPLAN              Catalog                 0          1          0
SYSIBM          SYSDBAUTH            Catalog                 0          3          0
```

The following is typical output resulting from a request for table space information:

```
             Tablespace Snapshot
First database connect timestamp           = 04-04-1997 14:29:55.197659
Last reset timestamp                       =
Snapshot timestamp                         = 04-04-1997 14:32:14.151875
Database name                              = SAMPLE
Database path                              = /home/user1/user1/NODE0000/SQL00011/
Input database alias                       = SAMPLE
Number of accessed tablespaces             = 3
Tablespace name                            = SYSCATSPACE
  Data pages copied to extended storage    = 0
  Index pages copied to extended storage   = 0
  Data pages copied from extended storage  = 0
  Index pages copied from extended storage = 0
  Buffer pool data logical reads           = 26
  Buffer pool data physical reads          = 11
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 55
  Buffer pool index physical reads         = 23
  Asynchronous pool index page reads       = 0
  Buffer pool index writes                 = 0
  Asynchronous pool index page writes      = 0
  Total buffer pool read time (ms)         = 342
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
  Asynchronous read requests               = 0
  Direct reads                             = 34
  Direct writes                            = 0
  Direct read requests                     = 4
  Direct write requests                    = 0
  Direct reads elapsed time (ms)           = 1
  Direct write elapsed time (ms)           = 0
  Number of files closed                   = 0
Tablespace name                            = TEMPSPACE1
  Data pages copied to extended storage    = 0
  Index pages copied to extended storage   = 0
  Data pages copied from extended storage  = 0
  Index pages copied from extended storage = 0
  Buffer pool data logical reads           = 0
  Buffer pool data physical reads          = 0
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 0
  Buffer pool index physical reads         = 0
  Asynchronous pool index page reads       = 0
  Buffer pool index writes                 = 0
  Asynchronous pool index page writes      = 0
  Total buffer pool read time (ms)         = 0
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
  Asynchronous read requests               = 0
  Direct reads                             = 0
  Direct writes                            = 0
  Direct read requests                     = 0
  Direct write requests                    = 0
  Direct reads elapsed time (ms)           = 0
  Direct write elapsed time (ms)           = 0
  Number of files closed                   = 0
Tablespace name                            = USERSPACE1
  Data pages copied to extended storage    = 0
  Index pages copied to extended storage   = 0
  Data pages copied from extended storage  = 0
  Index pages copied from extended storage = 0
  Buffer pool data logical reads           = 6
  Buffer pool data physical reads          = 2
  Asynchronous pool data page reads        = 0
  Buffer pool data writes                  = 0
  Asynchronous pool data page writes       = 0
  Buffer pool index logical reads          = 0
  Buffer pool index physical reads         = 0
  Asynchronous pool index page reads       = 0
  Buffer pool index writes                 = 0
  Asynchronous pool index page writes      = 0
  Total buffer pool read time (ms)         = 22
  Total buffer pool write time (ms)        = 0
  Total elapsed asynchronous read time     = 0
  Total elapsed asynchronous write time    = 0
  Asynchronous read requests               = 0
  Direct reads                             = 0
  Direct writes                            = 0
  Direct read requests                     = 0
  Direct write requests                    = 0
  Direct reads elapsed time (ms)           = 0
  Direct write elapsed time (ms)           = 0
  Number of files closed                   = 0
```

The following is typical output resulting from a request for lock information:

```
            Database Lock Snapshot
Database name                              = SAMPLE
Database path                              = /home/user1/user1/NODE0000/SQL00011/
Input database alias                       = SAMPLE
Locks held                                 = 7
Applications currently connected           = 1
Applications currently waiting on locks    = 0
Snapshot timestamp                         = 04-04-1997 14:32:14.151875
Application handle                         = 5
Application ID                             = *LOCAL.user1.970404192956
Sequence number                            = 0001
Application name                           = db2bp_32
Authorization ID                           = USER1
Application status                         = UOW Waiting
Status change time                         =
Application code page                    = 850
Locks held                               = 7
Total wait time (ms)                     = 0
Object  Object    Tablespace Name   Table Schema   Table Name     Mode Status
Name    Type
------- -------...---------------...------------...------------...---- -------
1545    Row       SYSCATSPACE       SYSIBM         SYSTABLES      NS   Granted
1544    Row       SYSCATSPACE       SYSIBM         SYSTABLES      NS   Granted
2       Table     SYSCATSPACE       SYSIBM         SYSTABLES      IS   Granted
27      Table     SYSCATSPACE       SYSIBM         SYSTABLESPACES S    Granted
257     Row       SYSCATSPACE       SYSIBM         SYSPLAN        S    Granted
7       Table     SYSCATSPACE       SYSIBM         SYSPLAN        IS   Granted
0       Internal                                                  S    Granted
```

The following is typical output resulting from a request for dynamic SQL information:

```
      Dynamic SQL Snapshot Result

 Database name                      = SAMPLE

 Database path                      = /home/smith/smith/NODE0000/SQL00001/

 Number of executions               = 2
 Number of compilations             = 1
 Worst preparation time (ms)        = 126
 Best preparation time (ms)         = 126
 Rows deleted                       = 0
 Rows inserted                      = 0
 Rows read                          = 24
 Rows updated                       = 0
 Rows written                       = 0
 Statement sorts                    = 0
 Total execution time (sec.ms)      = 0.060226
 Total system cpu time (sec.ms)     = 0
 Total user cpu time (sec.ms)       = 0
 Statement text                     = select * from org
```

## Usage Notes

To obtain a snapshot from a remote instance (or a different local instance), it is necessary to first attach to that instance. If an alias for a database residing at a different instance is specified, an error message is returned.

To obtain some statistics, it is necessary that the database system monitor switches are turned on.

No data is returned following a request for table information if any of the following is true:

- The TABLE recording switch is turned off.
- No tables have been accessed since the switch was turned on.

- No tables have been accessed since the last RESET MONITOR command was issued.

## See Also

"GET MONITOR SWITCHES" on page 252

"LIST APPLICATIONS" on page 295

"RESET MONITOR" on page 437.

## HELP

Permits the user to invoke help from the Information Center.

This command is not available on UNIX based systems.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──HELP──┬──────────────────┬──────────────────────────────────────────────►◄
          └─character-string─┘
```

### Command Parameters

**HELP character-string**
Any SQL or DB2 command, or any other item listed in the Information Center.

### Examples

Following are examples of the HELP command:

- `db2 help`

  This command opens the DB2 Information Center, which contains information about DB2 divided into categories, such as tasks, reference, books, and so on. This is equivalent to invoking the **db2ic** command with no parameters.
- `db2 help drop`

  This command opens the Web browser, and displays information about the SQL DROP statement. This is equivalent to invoking the following command: `db2ic -j drop`. The **db2ic** command searches first the *SQL Reference*, and then the *Command Reference*, for a statement or a command called DROP, and then displays the first one found.
- `db2 help 'drop database'`

  This command initiates a more refined search, and causes information about the DROP DATABASE command to be displayed.

## Usage Notes

The Information Center must be installed on the user's system. HTML books in the DB2 library must be located in the `\sqllib\doc\html` subdirectory.

The command line processor will not know if the command succeeds or fails, and cannot report error conditions.

## IMPORT

Inserts data from an external file with a supported file format into a table, hierarchy, or view. A faster alternative is "LOAD" on page 338; however, the load utility does not support loading data at the hierarchy level.

### Authorization

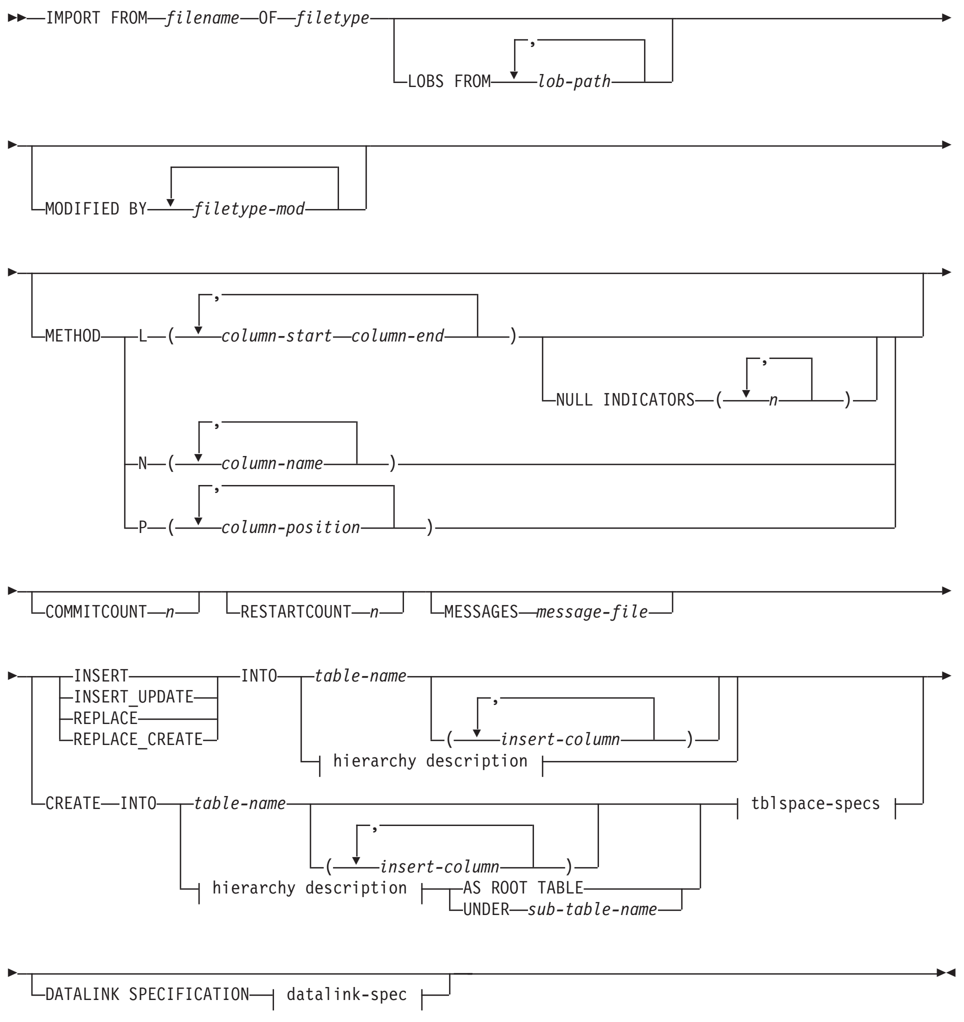
- IMPORT using the INSERT option requires one of the following:
  - *sysadm*
  - *dbadm*
  - CONTROL privilege on each participating table or view
  - INSERT and SELECT privilege on each participating table or view.
- IMPORT to an existing table using the INSERT_UPDATE, REPLACE, or the REPLACE_CREATE option, requires one of the following:
  - *sysadm*
  - *dbadm*
  - CONTROL privilege on the table or view.
- IMPORT to a table or a hierarchy that does not exist using the CREATE, or the REPLACE_CREATE option, requires one of the following:
  - *sysadm*
  - *dbadm*
  - CREATETAB authority on the database, and one of:
    - IMPLICIT_SCHEMA authority on the database, if the schema name of the table does not exist
    - CREATEIN privilege on the schema, if the schema of the table exists.
    - CONTROL privilege on every sub-table in the hierarchy, if the REPLACE_CREATE option on the entire hierarchy is used.
- IMPORT to an existing hierarchy using the REPLACE option requires one of the following:
  - *sysadm*
  - *dbadm*
  - CONTROL privilege on every sub-table in the hierarchy.

### Required Connection

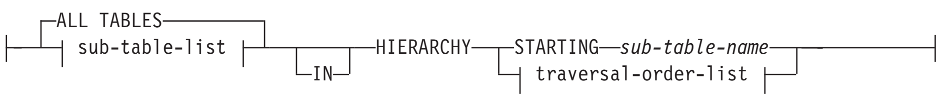
Database. If implicit connect is enabled, a connection to the default database is established.

## IMPORT

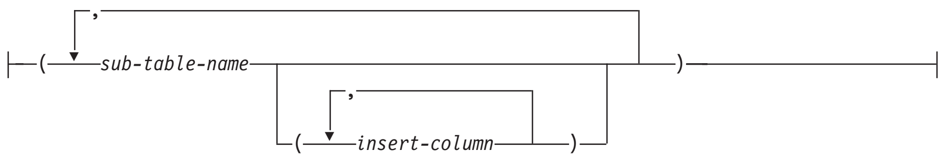
### Command Syntax



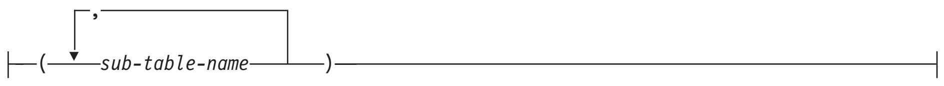
#### hierarchy description:



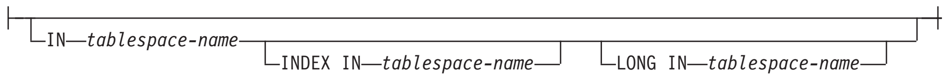
#### sub-table-list:



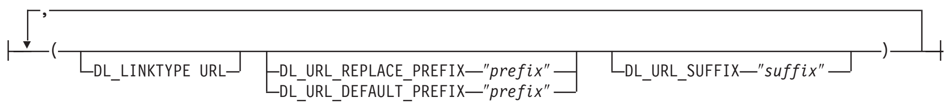
**traversal-order-list:**



**tblspace-specs:**



**datalink-spec:**


,
(
DL_LINKTYPE URL
DL_URL_REPLACE_PREFIX—"prefix"
DL_URL_DEFAULT_PREFIX—"prefix"
DL_URL_SUFFIX—"suffix"
)

## Command Parameters

**ALL TABLES**
An implicit keyword for hierarchy only. When importing a hierarchy, the default is to import all tables specified in the traversal order.

**AS ROOT TABLE**
Creates one or more sub-tables as a stand-alone table hierarchy.

**COMMITCOUNT n**
Performs a COMMIT after every *n* records are imported.

**CREATE**
Creates the table definition and row contents. If the data was exported from a DB2 table, sub-table, or hierarchy, indexes are created. If this option operates on a hierarchy, and data was exported from DB2, a type hierarchy will also be created. This option can only be used with IXF files.

**Note:** If the data was exported from an MVS host database, and it contains LONGVAR fields whose lengths, calculated on the page size, are less than 254, CREATE may fail because the rows are too long. In this case, the table should be created manually, and IMPORT with INSERT should be invoked, or, alternatively, the LOAD command should be used.

**DATALINK SPECIFICATION**

For each DATALINK column, there can be one column specification enclosed by parentheses. Each column specification consists of one or more DL_LINKTYPE, prefix, and a DL_URL_SUFFIX specification. The prefix specification can be either DL_URL_REPLACE_PREFIX or DL_URL_DEFAULT_PREFIX.

There can be as many DATALINK column specifications as the number of DATALINK columns defined in the table. The order of specifications follows the order of DATALINK columns found within the *insert-column* list, or within the table definition (if an *insert-column* list is not specified).

**DL_LINKTYPE**

If specified, it should match the LINKTYPE of the column definition. Thus, DL_LINKTYPE URL is acceptable if the column definition specifies LINKTYPE URL.

**DL_URL_DEFAULT_PREFIX** ″**prefix**″

If specified, it should act as the default prefix for all DATALINK values within the same column. In this context, prefix refers to the ″scheme host port″ part of the URL specification.

Examples of prefix are:

```
"http://server"
"file://server"
"file:"
"http://server:80"
```

If no prefix is found in a column’s data, and a default prefix is specified with DL_URL_DEFAULT_PREFIX, the default prefix is prefixed to the column value (if not NULL).

For example, if DL_URL_DEFAULT_PREFIX specifies the default prefix `"http://toronto"`:

- The column input value ″/x/y/z″ is stored as ″http://toronto/x/y/z″.
- The column input value ″http://coyote/a/b/c″ is stored as ″http://coyote/a/b/c″.
- The column input value NULL is stored as NULL.

**DL_URL_REPLACE_PREFIX** ″**prefix**″

This clause is useful for loading or importing data previously generated by the export utility, when the user wants to globally replace the host name in the data with another host name. If specified, it becomes the prefix for *all* non-NULL column values. If a column value has a prefix, this will replace it. If a column value has no prefix, the prefix specified by DL_URL_REPLACE_PREFIX is prefixed to the column value.

For example, if DL_URL_REPLACE_PREFIX specifies the prefix `"http://toronto"`:

- The column input value ″/x/y/z″ is stored as ″http://toronto/x/y/z″.
- The column input value ″http://coyote/a/b/c″ is stored as ″http://toronto/a/b/c″. Note that ″toronto″ replaces ″coyote″.
- The column input value NULL is stored as NULL.

**DL_URL_SUFFIX ″suffix″**
If specified, it is appended to every non-NULL column value for the column. It is, in fact, appended to the ″path″ component of the URL part of the DATALINK value.

**FROM filename**
Specifies the file that contains the data to be imported. If the path is omitted, the current working directory is used.

**HIERARCHY**
Specifies that hierarchical data is to be imported.

**IN tablespace-name**
Identifies the table space in which the table will be created. The table space must exist, and must be a REGULAR table space. If no other table space is specified, all table parts are stored in this table space. If this clause is not specified, the table is created in a table space created by the authorization ID. If none is found, the table is placed into the default table space USERSPACE1. If USERSPACE1 has been dropped, table creation fails.

**INDEX IN tablespace-name**
Identifies the table space in which any indexes on the table will be created. This option is allowed only when the primary table space specified in the IN clause is a DMS table space. The specified table space must exist, and must be a REGULAR DMS table space.

**Note:** Specifying which table space will contain an index can only be done when the table is created.

**insert-column**
Specifies the name of a column in the table or the view into which data is to be inserted.

**INSERT**
Adds the imported data to the table without changing the existing table data.

**INSERT_UPDATE**
Adds rows of imported data to the target table, or updates existing rows (of the target table) with matching primary keys.

**INTO table-name**
Specifies the database table into which the data is to be imported. This table cannot be a system table or a summary table.

One can use an alias for INSERT, INSERT_UPDATE, or REPLACE, except in the case of a down-level server, when the fully qualified or the unqualified table name should be used. A qualified table name is in the form: *schema.tablename*. The *schema* is the user name under which the table was created.

**LOBS FROM lob-path**
Specifies one or more paths that store LOB files. The names of the LOB data files are stored in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the LOB column. This option is ignored if the `lobsinfile` modifier is not specified.

**LONG IN tablespace-name**
Identifies the table space in which the values of any long columns (LONG VARCHAR, LONG VARGRAPHIC, LOB data types, or distinct types with any of these as source types) will be stored. This option is allowed only if the primary table space specified in the IN clause is a DMS table space. The table space must exist, and must be a LONG DMS table space.

**MESSAGES message-file**
Specifies the destination for warning and error messages that occur during an import operation. If the file already exists, the import utility appends the information. If the complete path to the file is not specified, the utility uses the current directory and the default drive as the destination. If *message-file* is omitted, the messages are written to standard output.

**METHOD**

**L** Specifies the start and end column numbers from which to import data.

**Note:** This method can only be used with ASC files, and is the only valid option for that file type.

**N** Specifies the names of the columns to be imported.

**Note:** This method can only be used with IXF files.

**P** Specifies the numbers of the columns to be imported.

**Note:** This method can only be used with IXF or DEL files, and is the only valid option for the DEL file type.

**MODIFIED BY filetype-mod**
Specifies additional options (see Table 6 on page 285).

**NULL INDICATORS n**
Specifies (by number) one or more columns in the data file that are to be used as null indicator fields. If this option is used, a null indicator column for each data column must be specified. Zero (`0`) indicates that the data column is not nullable, and that there will always be data in that column.

While processing each row, a `Y` indicates that the column data is NULL, while an `N` indicates that the column data is not NULL, and that column data specified by the METHOD L option will be imported.

**OF filetype**
Specifies the format of the data in the input file:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format), which is used by a variety of database manager and file manager programs
- WSF (work sheet format), which is used by programs such as:
  - Lotus 1-2-3
  - Lotus Symphony
- IXF (integrated exchange format, PC version), which means it was exported from the same or another DB2 table. An IXF file also contains the table definition and definitions of any existing indexes, except when columns are specified in the SELECT statement.

For more information about file formats, see the "Export/Import/Load Utility File Formats" appendix in the *Data Movement Utilities Guide and Reference.*

**REPLACE**
Deletes all existing data from the table by truncating the data object, and inserts the imported data. The table definition and the index definitions are not changed. This option can only be used if the table exists. It is not valid for tables with DATALINK columns. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

**REPLACE_CREATE**
If the table exists, deletes all existing data from the table by truncating the data object, and inserts the imported data without changing the table definition or the index definitions.

If the table does not exist, creates the table and index definitions, as well as the row contents.

This option can only be used with IXF files. It is not valid for tables with DATALINK columns. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

**RESTARTCOUNT n**
Specifies that an import operation is to be started at record *n* + 1. The first *n* records are skipped.

**STARTING sub-table-name**
A keyword for hierarchy only, requesting the default order, starting from *sub-table-name*. For PC/IXF files, the default order is the order stored in the input file. The default order is the only valid order for the PC/IXF file format.

**sub-table-list**
For typed tables with the INSERT or the INSERT_UPDATE option, a list of sub-table names is used to indicate the sub-tables into which data is to be imported.

**traversal-order-list**
For typed tables with the INSERT, INSERT_UPDATE, or the REPLACE option, a list of sub-table names is used to indicate the traversal order of the importing sub-tables in the hierarchy.

**UNDER sub-table-name**
Specifies a parent table for creating one or more sub-tables.

## Examples

The following example shows how to import information from `myfile.ixf` to the STAFF table:

```
db2 import from myfile.ixf of ixf messages msg.txt insert into staff
```

```
SQL3150N  The H record in the PC/IXF file has product "DB2    01.00", date
"19970220", and time "140848".

SQL3153N  The T record in the PC/IXF file has name "myfile", qualifier "        ",
and source "            ".

SQL3109N  The utility is beginning to load data from file "myfile".

SQL3110N  The utility has completed processing.  "58" rows were read from the
input file.

SQL3221W  ...Begin COMMIT WORK. Input Record Count = "58".

SQL3222W  ...COMMIT of any database changes was successful.

SQL3149N  "58" rows were processed from the input file.  "58" rows were
successfully inserted into the table.  "0" rows were rejected.
```

The following example shows how to import the table MOVIETABLE from the input file delfile1, which has data in the DEL format:

```
   db2 import from delfile1 of del
       modified by dldel|
       insert into movietable (actorname, description, url_making_of, url_movie)
       datalink specification (dl_url_default_prefix "http://narang"),
       (dl_url_replace_prefix "http://bomdel" dl_url_suffix ".mpeg")
```

**Notes:**

1. The table has four columns:

```
actorname              VARCHAR(n)
description            VARCHAR(m)
url_making_of          DATALINK (with LINKTYPE URL)
url_movie              DATALINK (with LINKTYPE URL)
```

2. The DATALINK data in the input file has the vertical bar (|) character as the sub-field delimiter.
3. If any column value for url_making_of does not have the prefix character sequence, ″http://narang″ is used.
4. Each non-NULL column value for url_movie will get ″http://bomdel″ as its prefix. Existing values are replaced.
5. Each non-NULL column value for url_movie will get ″.mpeg″ appended to the path. For example, if a column value of url_movie is ″http://server1/x/y/z″, it will be stored as ″http://bomdel/x/y/z.mpeg″; if the value is ″/x/y/z″, it will be stored as ″http://bomdel/x/y/z.mpeg″.

## Usage Notes

Be sure to complete all table operations and release all locks before starting an import operation. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.

The import utility adds rows to the target table using the SQL INSERT statement. The utility issues one INSERT statement for each row of data in the input file. If an INSERT statement fails, one of two actions result:

- If it is likely that subsequent INSERT statements can be successful, a warning message is written to the message file, and processing continues.
- If it is likely that subsequent INSERT statements will fail, and there is potential for database damage, an error message is written to the message file, and processing halts.

The utility performs an automatic COMMIT after the old rows are deleted during a REPLACE or a REPLACE_CREATE operation. Therefore, if the system fails, or the application interrupts the database manager after the table object is truncated, all of the old data is lost. Ensure that the old data is no longer needed before using these options.

If the log becomes full during a CREATE, REPLACE, or REPLACE_CREATE operation, the utility performs an automatic COMMIT on inserted records. If the system fails, or the application interrupts the database manager after an automatic COMMIT, a table with partial data remains in the database. Use the REPLACE or the REPLACE_CREATE option to rerun the whole import operation, or use INSERT with the RESTARTCOUNT parameter set to the number of rows successfully imported.

By default, automatic COMMITs are not performed for the INSERT or the INSERT_UPDATE option. They are, however, performed if the COMMITCOUNT parameter is not zero. A full log results in a ROLLBACK.

Whenever the import utility performs a COMMIT, two messages are written to the message file: one indicates the number of records to be committed, and the other is written after a successful COMMIT. When restarting the import operation after a failure, specify the number of records to skip, as determined from the last successful COMMIT.

The import utility accepts input data with minor incompatibility problems (for example, character data can be imported using padding or truncation, and numeric data can be imported with a different numeric data type), but data with major incompatibility problems is not accepted.

One cannot REPLACE or REPLACE_CREATE an object table if it has any dependents other than itself, or an object view if its base table has any dependents (including itself). To replace such a table or a view, do the following:

1. Drop all foreign keys in which the table is a parent.
2. Run the import utility.
3. Alter the table to recreate the foreign keys.

If an error occurs while recreating the foreign keys, modify the data to maintain referential integrity.

Referential constraints and foreign key definitions are not preserved when creating tables from PC/IXF files. (Primary key definitions *are* preserved if the data was previously exported using `SELECT *`.)

Importing to a remote database requires enough disk space on the server for a copy of the input data file, the output message file, and potential growth in the size of the database.

If an import operation is run against a remote database, and the output message file is very long (more than 60KB), the message file returned to the user on the client may be missing messages from the middle of the import operation. The first 30KB of message information and the last 30KB of message information are always retained.

Importing PC/IXF files to a remote database is much faster if the PC/IXF file is on a hard drive rather than on diskettes.

The database table or hierarchy must exist before data in the ASC, DEL, or WSF file formats can be imported; however, if the table does not already exist, IMPORT CREATE or IMPORT REPLACE_CREATE creates the table when it imports data from a PC/IXF file. For typed tables, IMPORT CREATE can create the type hierarchy and the table hierarchy as well.

PC/IXF import should be used to move data (including hierarchical data) between databases. If character data containing row separators is exported to a delimited ASCII (DEL) file and processed by a text transfer program (moving, for example between OS/2 and AIX systems), fields containing the row separators will shrink or expand. PC/IXF file format specifications permit migration of data between OS/2 (IBM Extended Services for OS/2, OS/2 Extended Edition, and DB2 for OS/2) databases and DB2 for AIX databases via export, binary copying of files between OS/2 and AIX, and import. The file copying step is not necessary if the source and the target databases are both accessible from the same client.

The data in ASC and DEL files is assumed to be in the code page of the client application performing the import. PC/IXF files, which allow for different code pages, are recommended when importing data in different code pages. If the PC/IXF file and the import utility are in the same code page, processing occurs as for a regular application. If the two differ, and the FORCEIN option is specified, the import utility assumes that data in the PC/IXF file has the same code page as the application performing the import. This occurs even if there is a conversion table for the two code pages. If the two differ, the FORCEIN option is not specified, and there is a conversion table, all data in the PC/IXF file will be converted from the file code page to the application code page. If the two differ, the FORCEIN option is not specified, and there is no conversion table, the import operation will fail. This applies only to PC/IXF files on DB2 for AIX clients.

For table objects on an 8KB page that are close to the limit of 1012 columns, import of PC/IXF data files may cause DB2 to return an error, because the maximum size of an SQL statement was exceeded. This situation can occur only if the columns are of type CHAR, VARCHAR, or CLOB. The restriction does not apply to import of DEL or ASC files. If PC/IXF files are being used to create a new table, an alternative is to dump the source table's DDL using "db2look - DB2 Statistics Extraction Tool" on page 52, and then to issue that statement through the CLP.

DB2 Connect can be used to import data to DRDA servers such as DB2 for OS/390, DB2 for VM and VSE, and DB2 for OS/400. Only PC/IXF import (INSERT option) is supported. The RESTARTCOUNT parameter, but not the COMMITCOUNT parameter, is also supported.

When using the CREATE option with typed tables, create every sub-table defined in the PC/IXF file; sub-table definitions cannot be altered. When using options other than CREATE with typed tables, the traversal order list enables one to specify the traverse order; therefore, the traversal order list must match the one used during the export operation. For the PC/IXF file format, one need only specify the target sub-table name, and use the traverse order stored in the file.

The import utility can be used to recover a table previously exported to a PC/IXF file. The table returns to the state it was in when exported.

Data cannot be imported to a system table or a summary table.

Views cannot be created through the import utility.

Importing a multiple-part PC/IXF file whose individual parts are copied from an OS/2 system to an AIX system is supported on DB2.

On the Windows NT operating system:

- Importing logically split PC/IXF files is not supported.
- Importing bad format PC/IXF or WSF files is not supported.

**DB2 Data Links Manager Considerations**

Before running the DB2 import utility, do the following:

1. Copy the files that will be referenced to the appropriate Data Links servers. The **dlfm_import** utility can be used to extract files from an archive that is generated by the **dlfm_export** utility.
2. Register the required prefix names to the DB2 Data Links Managers. There may be other administrative tasks, such as registering the database, if required.
3. Update the Data Links server information in the URLs (of the DATALINK columns) from the exported data for the SQL table, if required. (If the original configuration's Data Links servers are the same at the target location, the Data Links server names need not be updated.)
4. Define the Data Links servers at the target configuration in the DB2 Data Links Manager configuration file.

When the import utility is executed on the target system, data related to DATALINK columns is loaded into the underlying DB2 tables using SQL INSERT (as is the case for other columns).

During the insert operation, DATALINK column processing links the files in the appropriate Data Links servers according to the column specifications at the target database.

**Representation of DATALINK Information in an Input File**

For a description of how DATALINK information is represented in an input file, see page 356.

Table 6. Valid File Type Modifiers (Import)

| **Modifier** | **Description** |
|---|---|
| **All File Formats** | |
| compound=*x* | *x* is a number between 1 and 100 inclusive (7 on DOS/Windows). Uses nonatomic compound SQL to insert the data, and *x* statements will be attempted each time. |
| lobsinfile | *lob-path* specifies the path to the files containing LOB values. |
| no_type_id | Valid only when importing into a single sub-table. Typical usage is to export data from a regular table, and then to invoke an import operation (using this modifier) to convert the data into a single sub-table. |

Table 6. Valid File Type Modifiers (Import) (continued)

| **Modifier** | **Description** |
|---|---|
| nodefaults | If a source column for a target table column is not explicitly specified, and the table column is not nullable, default values are not loaded. Without this option, if a source column for one of the target table columns is not explicitly specified, one of the following occurs:<br>• If the column is defaultable, the default value is loaded<br>• If the column is nullable and not defaultable, a NULL is loaded<br>• If the column is not nullable and not defaultable, an error is returned, and the utility stops processing. |
| usedefaults | If a source column for a target table column has been specified, but it contains no data for one or more row instances, default values are loaded. Examples of missing data are:<br>• For DEL files: ",," is specified for the column<br>• For ASC files: The NULL indicator is set to yes for the column<br>• For DEL/ASC/WSF files: A row that does not have enough columns, or is not long enough for the original specification.<br><br>Without this option, if a source column contains no data for a row instance, one of the following occurs:<br>• If the column is nullable, a NULL is loaded<br>• If the column is not nullable, the utility rejects the row. |
| **ASCII File Formats (ASC/DEL)** | |
| implieddecimal | The location of an implied decimal point is determined by the column definition; it is no longer assumed to be at the end of the value. For example, the value `12345` is loaded into a DECIMAL(8,2) column as `123.45`, *not* `12345.00`. |
| noeofchar | The optional end-of-file character `x'1A'` is not recognized as the end of file. Processing continues as if it were a normal character. |
| **ASC (Non-delimited ASCII) File Format** | |

Table 6. Valid File Type Modifiers (Import) (continued)

| **Modifier** | **Description** |
|---|---|
| nochecklengths | If `nochecklengths` is specified, an attempt is made to import each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully imported if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions. |
| nullindchar=*x* | *x* is a single character. Changes the character denoting a null value to *x*. The default value of *x* is `Y`.[b]<br><br>This modifier is case sensitive for EBCDIC data files, except when the character is an English letter. For example, if the null indicator character is specified to be the letter `N`, then `n` is also recognized as a null indicator. |
| reclen=*x* | *x* is an integer with a maximum value of 32 767. *x* characters are read for each row, and a new-line character is not used to indicate the end of the row. |
| striptblanks | Truncates any trailing blank spaces when loading data into a variable-length field. If this option is not specified, blank spaces are kept.<br><br>In the following example, `striptblanks` causes the import utility to truncate trailing blank spaces:<br><br>`db2 import from myfile.asc of asc`<br>`   modified by striptblanks`<br>`   method l (1 10, 12 15) messages msgs.txt`<br>`   insert into staff`<br><br>This option cannot be specified together with `striptnulls`. These are mutually exclusive options.<br>**Note:** This option replaces the obsolete `t` option, which is supported for back-level compatibility only. |
| striptnulls | Truncates any trailing NULLs (0x00 characters) when loading data into a variable-length field. If this option is not specified, NULLs are kept.<br><br>This option cannot be specified together with `striptblanks`. These are mutually exclusive options.<br>**Note:** This option replaces the obsolete `padwithzero` option, which is supported for back-level compatibility only. |
| **DEL (Delimited ASCII) File Format** | |

Table 6. Valid File Type Modifiers (Import) (continued)

| **Modifier** | **Description** |
|---|---|
| chardel*x* | *x* is a single character string delimiter. The default value is a double quotation mark ("). The specified character is used in place of double quotation marks to enclose a character string.[ab]<br><br>The single quotation mark (') can also be specified as a character string delimiter. In the following example, `chardel''` causes the import utility to interpret any single quotation mark (') it encounters as a character string delimiter:<br>`db2 "import from myfile.del of del`<br>`   modified by chardel''`<br>`   method p (1, 4) insert into staff (id, years)"` |
| coldel*x* | *x* is a single character column delimiter. The default value is a comma (,). The specified character is used in place of a comma to signal the end of a column.[ab]<br><br>In the following example, `coldel;` causes the import utility to interpret any semicolon (;) it encounters as a column delimiter:<br>`db2 import from myfile.del of del`<br>`   modified by coldel;`<br>`   messages msgs.txt insert into staff` |
| datesiso | Date format. Causes all date data values to be imported in ISO format. |
| decplusblank | Plus sign character. Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign. |
| decpt*x* | *x* is a single character substitute for the period as a decimal point character. The default value is a period (.). The specified character is used in place of a period as a decimal point character.[ab]<br><br>In the following example, `decpt;` causes the import utility to interpret any semicolon (;) it encounters as a decimal point:<br>`db2 "import from myfile.del of del`<br>`   modified by chardel'`<br>`   decpt; messages msgs.txt insert into staff"` |

Table 6. Valid File Type Modifiers (Import) (continued)

| **Modifier** | **Description** |
|---|---|
| delprioritychar | The current default priority for delimiters is: record delimiter, character delimiter, column delimiter. This modifier protects existing applications that depend on the older priority by reverting the delimiter priorities to: character delimiter, record delimiter, column delimiter. Syntax:<br>`db2 import ... modified by delprioritychar ...`<br><br>For example, given the following DEL data file:<br>`"Smith, Joshua",4000,34.98<row delimiter>`<br>`"Vincent,<row delimiter>, is a manager", ...`<br>`... 4005,44.37<row delimiter>`<br><br>With the `delprioritychar` modifier specified, there will be only two rows in this data file. The second <row delimiter> will be interpreted as part of the first data column of the second row, while the first and the third <row delimiter> are interpreted as actual record delimiters. If this modifier is *not* specified, there will be three rows in this data file, each delimited by a <row delimiter>. |
| dldel*x* | *x* is a single character DATALINK delimiter. The default value is a semicolon (;). The specified character is used in place of a semicolon as the inter-field separator for a DATALINK value. It is needed because a DATALINK value may have more than one sub-value. [ab]<br>**Note:** *x* must not be the same character specified as the row, column, or character string delimiter. |
| nodoubledel | Suppresses recognition of double character delimiters. |
| **IXF File Format** | |
| forcein | Directs the utility to accept data despite code page mismatches, and to suppress translation between code pages.<br><br>Fixed length target fields are checked to verify that they are large enough for the data. If `nochecklengths` is specified, no checking is done, and an attempt is made to import each row. |
| indexixf | Directs the utility to drop all indexes currently defined on the existing table, and to create new ones from the index definitions in the PC/IXF file. This option can only be used when the contents of a table are being replaced. It cannot be used with a view, or when a *insert-column* is specified. |

Table 6. Valid File Type Modifiers (Import) (continued)

| **Modifier** | **Description** |
|---|---|
| indexschema=*schema* | Uses the specified *schema* for the index name during index creation. If *schema* is not specified (but the keyword `indexschema` *is* specified), uses the connection user ID. If the keyword is not specified, uses the schema in the IXF file. |
| nochecklengths | If `nochecklengths` is specified, an attempt is made to import each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully imported if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions. |

**Notes:**

1. The import utility does not issue a warning if an attempt is made to use unsupported file types with the MODIFIED BY option. If this is attempted, the import operation fails, and an error code is returned.
2. [a] "Delimiter Restrictions" on page 213 lists restrictions that apply to the characters that can be used as delimiter overrides.
3. [b] The character must be specified in the code page of the source data.

   The character code point (instead of the character symbol), can be specified using the syntax xJJ or 0xJJ, where JJ is the hexadecimal representation of the code point. For example, to specify the # character as a column delimiter, use one of the following:

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```

## See Also

"EXPORT" on page 206

"LOAD" on page 338.

## INITIALIZE TAPE

DB2 for Windows NT supports backup and restore to streaming tape devices. Use this command for tape initialization.

This command is available on Windows NT only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──INITIALIZE TAPE──┬─────────────┬──┬──────────────────┬──────────────────────►◄
                     └─ON──device──┘  └─USING──blksize──┘
```

### Command Parameters

**ON device**
Specifies a valid tape device name. The default value is `\\.\TAPE0`.

**USING blksize**
Specifies the block size for the device, in bytes. The device is initialized to use the block size specified, if the value is within the supported range of block sizes for the device.

**Note:** The buffer size specified in “BACKUP DATABASE” on page 124, and in “RESTORE DATABASE” on page 441 must be divisible by the block size specified here.

If a value for this parameter is not specified, the device is initialized to use its default block size. If a value of zero is specified, the device is initialized to use a variable length block size; if the device does not support variable length block mode, an error is returned.

### See Also

“REWIND TAPE” on page 450

“SET TAPE POSITION” on page 474.

## INVOKE STORED PROCEDURE

Invokes a procedure stored at the location of a database. Also known as the Database Application Remote Interface (DARI). The server procedure executes at the location of the database, and returns data to the client application.

The application programmer designs the program to run in two parts, one on the client and the other on the server. The server procedure at the database runs within the same transaction as the client application. If the client application and the server procedure are on the same node, the server procedure is executed locally.

**Note:** This command has been replaced by the SQL CALL statement (see the *SQL Reference*). SQL CALL cannot be used from within the CLP.

### Authorization

CONNECT privilege on a database.

### Required Connection

Database

### Command Syntax

```
►►──INVOKE──program-name──┬────────────────────────────┬──────────────────────►◄
                          └─USING──server–input–data───┘
```

**Note:** Do not use INVOKE to call server procedures that use input or output SQLDA structures, including server procedures that return data. For more information, see the *Application Development Guide*.

### Command Parameters

**program-name**
Specifies the procedure to be run on the server. This parameter can be specified in one of the following ways:

- By a *procName* with no extensions. This notation tells the database manager to load a DARI library named *procName* into memory. It is assumed that the name of the function routine to be executed is identical to the library name. For example,

  ```
  db2 invoke foo
  ```

  will load the DARI library named `FOO` and execute the function routine `FOO()` within the library.

The database manager will find the DARI libraries in the default directory `$HOME/sqllib/function` of the instance owner.

- By the exclamation sign (!) delimited name, as in *procName!funcName*. This notation tells the database manager to load a DARI library, named `procName`, into memory. The function routine to be executed is `funcName`. This convention allows similar function routines to be packaged in the same DARI library.
- By an absolute path name, as in `/u/cche/procName!/funcName`, for example. This notation includes the storage path of the DARI library. In this example, the DARI library named `procName` is stored in the directory `/u/cche`. The function routine to be executed is `funcName`.

**Note:** To support portability between various versions of DB2 products, the `!` delimiter can be replaced by the backslash (`\`) delimiter.

**USING server-input-data**
Specifies any information that is passed to the server routine. A variable character string, free form, flexible parameter that can be used to transmit input data according to specific needs.

## LIST ACTIVE DATABASES

Displays a subset of the information listed by the GET SNAPSHOT FOR ALL DATABASES command (see “GET SNAPSHOT” on page 254). For each active database, this command displays the following:

- Database name
- Number of applications currently connected to the database
- Database path.

### Scope

This command can be issued from any node that is listed in `$HOME/sqllib/db2nodes.cfg`. It returns the same information from any of these nodes.

### Authorization

None

### Command Syntax

```
►►──LIST ACTIVE DATABASES──────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

Following is sample output from the LIST ACTIVE DATABASES command:

```
                           Active Databases

Database name                           = TEST
Applications connected currently        = 0
Database path                           = /home/smith/smith/NODE0000/SQL00002/

Database name                           = SAMPLE
Applications connected currently        = 1
Database path                           = /home/smith/smith/NODE0000/SQL00001/
```

### See Also

“GET SNAPSHOT” on page 254.

## LIST APPLICATIONS

Displays to standard output the application program name, authorization ID (user name), application handle, application ID, and database name of all active database applications. This command can also optionally display an application’s sequence number, status, status change time, and database path.

### Scope

This command only returns information for the node on which it is issued.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance. To list applications for a remote instance, it is necessary to first attach to that instance.

### Command Syntax

```
►►──LIST APPLICATIONS──┬─────────────────────────────────────┬──┬─────────────┬──────►◄
                       └─FOR──┬─DATABASE─┬──database-alias─┘  └─SHOW DETAIL─┘
                              └─DB───────┘
```

### Command Parameters

**FOR DATABASE database-alias**
Information for each application that is connected to the specified database is to be displayed. Database name information is not displayed. If this option is not specified, the command displays the information for each application that is currently connected to any database at the node to which the user is currently attached.

The default application information is comprised of the following:
- Authorization ID
- Application program name
- Application handle
- Application ID
- Database name.

**SHOW DETAIL**
Output will include the following additional information:
- Sequence #
- Application status
- Status change time
- Database path.

**Note:** If this option is specified, it is recommended that the output be redirected to a file, and that the report be viewed with the help of an editor. The output lines may wrap around when displayed on the screen.

## Examples

The following is sample output from LIST APPLICATIONS:

```
Auth Id  Application    Appl.      Application Id                 DB       # of
         Name           Handle                                    Name    Agents
-------- -------------- ---------- ------------------------------ -------- -----
smith    db2bp_32       12         *LOCAL.smith.970220191502      TEST     1
smith    db2bp_32       11         *LOCAL.smith.970220191453      SAMPLE   1
```

**Note:** For more information about these fields, see the *System Monitor Guide and Reference*.

## Usage Notes

The database administrator can use the output from this command as an aid to problem determination. In addition, this information is required if the database administrator wants to use “GET SNAPSHOT” on page 254 or “FORCE APPLICATION” on page 215 in an application.

To list applications at a remote instance (or a different local instance), it is necessary to first attach to that instance. If FOR DATABASE is specified when an attachment exists, and the database resides at an instance which differs from the current attachment, the command will fail.

## LIST COMMAND OPTIONS

Lists the current settings for the environment variables:
- **DB2BQTIME**
- **DB2DQTRY**
- **DB2RQTIME**
- **DB2IQTIME**
- **DB2OPTIONS**.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──LIST COMMAND OPTIONS──────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

The following is sample output from LIST COMMAND OPTIONS:

```
     Command Line Processor Option Settings

 Backend process wait time (seconds)        (DB2BQTIME) = 1
 No. of retries to connect to backend        (DB2BQTRY) = 60
 Request queue wait time (seconds)          (DB2RQTIME) = 5
 Input queue wait time (seconds)            (DB2IQTIME) = 5
 Command options                           (DB2OPTIONS) =

 Option  Description                                 Current Setting
 ------  ----------------------------------------    ---------------
   -a    Display SQLCA                               OFF
   -c    Auto-Commit                                 ON
   -e    Display SQLCODE/SQLSTATE                    OFF
   -f    Read from input file                        OFF
   -l    Log commands in history file                OFF
   -o    Display output                              ON
   -p    Display interactive input prompt            ON
   -r    Save output to report file                  OFF
   -s    Stop execution on command error             OFF
   -t    Set statement termination character         OFF
   -v    Echo current command                         OFF
   -w    Display FETCH/SELECT warning messages        ON
   -z    Save all output to output file               OFF
```

## Usage Notes

For detailed information about these options, see "Command Line Processor Options" on page 97.

## LIST DATABASE DIRECTORY

Lists the contents of the system database directory. If a path is specified, the contents of the local database directory are listed.

### Scope

If this command is issued without the ON *path* parameter, the system database directory is returned. This information is the same at all nodes.

If the ON *path* parameter is specified, the local database directory on that path is returned. This information is not the same at all nodes.

### Authorization

None

### Required Connection

None. Directory operations affect the local directory only.

### Command Syntax

```
►►──LIST──┬─DATABASE─┬──DIRECTORY──┬─────────────────┬──────────────────────────────►◄
          └─DB───────┘             └─ON──┬─path──┬─┘
                                         └─drive─┘
```

### Command Parameters

**ON path/drive**
: Specifies the local database directory from which to list information. If not specified, the contents of the system database directory are listed.

### Examples

The following shows sample output for a system database directory:

```
 System Database Directory

 Number of entries in the directory = 2

Database 1 entry:

 Database alias                   = SAMPLE
 Database name                    = SAMPLE
 Database drive                   = /home/smith
 Database release level           = 8.00
 Comment                          =
 Directory entry type             = Indirect
 Catalog node number              = 0
```

```
Database 2 entry:

 Database alias                   = GDB1
 Database global name             = /.../cell_name/dir_name/gdb1
 Database release level           = 8.00
 Comment                          = DCE database
 Directory entry type             = DCE
 Catalog node number              = -1
```

The following shows sample output for a local database directory:

```
 Local Database Directory on /u/smith

 Number of entries in the directory = 1

Database 1 entry:

 Database alias                   = SAMPLE
 Database name                    = SAMPLE
 Database directory               = SQL00001
 Database release level           = 8.00
 Comment                          =
 Directory entry type             = Home
 Catalog node number              = 0
 Node number                      = 0
```

These fields are identified as follows:

**Database alias**
: The value of the *alias* parameter when the database was created or cataloged. If an alias was not entered when the database was cataloged, the database manager uses the value of the *database-name* parameter when the database was cataloged.

**Database global name**
: The fully qualified name that uniquely identifies the database in the DCE name space.

**Database name**
: The value of the *database-name* parameter when the database was cataloged. This name is usually the name under which the database was created.

**Local database directory**
: The path on which the database resides. This field is filled in only if the system database directory has been scanned.

**Database directory/Database drive**
: The name of the directory or drive where the database resides. This field is filled in only if the local database directory has been scanned.

**Node name**
The name of the remote node. This name corresponds to the value entered for the *nodename* parameter when the database and the node were cataloged.

**Database release level**
The release level of the database manager that can operate on the database.

**Comment**
Any comments associated with the database that were entered when it was cataloged.

**Directory entry type**
The location of the database:

- A *remote* entry describes a database that resides on another node.
- An *indirect* entry describes a database that is local. Databases that reside on the same node as the system database directory are thought to indirectly reference the home entry (to a local database directory), and are considered indirect entries.
- A *home* entry indicates that the database directory is on the same path as the local database directory.

All entries in the system database directory are either remote or indirect. All entries in local database directories are identified in the system database directory as indirect entries.

**Authentication**
The authentication type cataloged at the client is used to determine whether the connection is being done with system or with DCE security.

**Catalog node number**
Specifies which node is the catalog node (MPP systems only). This is the node on which the CREATE DATABASE command was issued.

**Node number**
Specifies the number that is assigned in `db2nodes.cfg` to the node where the command was issued (MPP systems only). In non-MPP systems where there is no `db2nodes.cfg` file, the node number will always be zero.

## Usage Notes

There can be a maximum of eight opened database directory scans per process. To overcome this restriction for a batch file that issues more than eight LIST DATABASE DIRECTORY commands within a single DB2 session, convert the batch file into a shell script. The ″db2″ prefix generates a new DB2 session for each command.

### See Also

"CHANGE DATABASE COMMENT" on page 183

"CREATE DATABASE" on page 187.

## LIST DCS APPLICATIONS

Displays to standard output information about applications that are connected to host databases via DB2 Connect Enterprise Edition.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance. To list the DCS applications at a remote instance, it is necessary to first attach to that instance.

### Command Syntax

```
►►──LIST DCS APPLICATIONS──┬─────────────┬──────────────────────────────────►◄
                           ├─SHOW DETAIL─┤
                           └─EXTENDED────┘
```

### Command Parameters

**LIST DCS APPLICATIONS**
The default application information includes:
- Host authorization ID (*username*)
- Application program name
- Application handle
- Outbound application ID (*luwid*).

**SHOW DETAIL**
Specifies that output include the following additional information:
- Client application ID
- Client sequence number
- Client database alias
- Client node name (*nname*)
- Client release level
- Client code page
- Outbound sequence number
- Host database name
- Host release level.

**EXTENDED**
Generates an extended report. This report includes all of the fields that are listed when the SHOW DETAIL option is specified, plus the following additional fields:

- DCS application status
- Status change time
- Client platform
- Client protocol
- Client code page
- Process ID of the client application
- Host coded character set ID (CCSID).

## Examples

The following is sample output from LIST DCS APPLICATIONS:

```
   Auth Id  Application Name     Appl.      Outbound Application Id
                                 Handle
   -------- -------------------- ---------- --------------------------------
   DDCSUS1  db2bp_s              2          0915155C.139D.971205184245
```

The following is sample output from LIST DCS APPLICATIONS EXTENDED:

```
               List of DCS Applications - Extended Report

Client application ID                     = 09151251.0AD1.980529194106
  Sequence number                         = 0001
  Authorization ID                        = SMITH
  Application name                        = db2bp
  Application handle                      = 0
  Application status                      = waiting for reply
  Status change time                      = Not Collected
  Client DB alias                         = MVSDB
  Client node                             = antman
  Client release level                    = SQL05020
  Client platform                         = AIX
  Client protocol                         = TCP/IP
  Client codepage                         = 819
  Process ID of client application        = 38340
  Client login ID                         = user1
  Host application ID                     = G9151251.GAD2.980529194108
  Sequence number                         = 0000
  Host DB name                            = GILROY
  Host release level                      = DSN05011
  Host CCSID                              = 500
```

**Notes:**

1. The application status field contains one of the following values:

**connect pending - outbound**
Denotes that the request to connect to a host database has been issued, and that DB2 Connect is waiting for the connection to be established.

**waiting for request**
Denotes that the connection to the host database has been established, and that DB2 Connect is waiting for an SQL statement from the client application.

**waiting for reply**
Denotes that the SQL statement has been sent to the host database.

2. The status change time is shown only if the System Monitor UOW switch was turned on during processing. Otherwise, `Not Collected` is shown.
3. For more information about these fields, see the *System Monitor Guide and Reference*.

## Usage Notes

The database administrator can use this command to match client application connections *to* the gateway with corresponding host connections *from* the gateway.

The database administrator can also use agent ID information to force specified applications off a DB2 Connect server.

## See Also

"FORCE APPLICATION" on page 215.

## LIST DCS DIRECTORY

Lists the contents of the Database Connection Services (DCS) directory.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──LIST DCS DIRECTORY──────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

The following is sample output from LIST DCS DIRECTORY:

```
 Database Connection Services (DCS) Directory

 Number of entries in the directory = 1

DCS 1 entry:

 Local database name                = DB2
 Target database name               = DSN_DB_1
 Application requestor name         =
 DCS parameters                     =
 Comment                            = DB2/MVS Location name DSN_DB_1
 DCS directory release level        = 0x0100
```

These fields are identified as follows:

**Local database name**
Specifies the alias of the target host database. This corresponds to the *database-name* parameter entered when the host database was cataloged in the DCS directory.

**Target database name**
Specifies the name of the host database that can be accessed. This corresponds to the *target-database-name* parameter entered when the host database was cataloged in the DCS directory.

**Application requester name**
Specifies the name of the program residing on the application requester or server.

**DCS parameters**
String that contains the connection and operating environment parameters to use with the application requester. Corresponds to the parameter string entered when the host database was cataloged. The string must be enclosed by double quotation marks, and the parameters must be separated by commas.

For more information about DCS parameters, see the *DB2 Connect User's Guide*.

**Comment**
Describes the database entry.

**DCS directory release level**
Specifies the version number of the Distributed Database Connection Services program under which the database was created.

## Usage Notes

The DCS directory is created the first time that "CATALOG DCS DATABASE" on page 157 is invoked. It is maintained on the path / drive where DB2 was installed, and provides information about host databases that the workstation can access if the DB2 Connect program has been installed. The host databases can be:

- DB2 for MVS / ESA databases on System / 370 and System / 390 architecture host computers
- DB2 for VSE & VM databases on System / 370 and System / 390 architecture host computers
- DB2 Universal Database for AS / 400 databases on Application System / 400 (AS / 400) host computers.

## LIST DRDA INDOUBT TRANSACTIONS

Provides a list of transactions that are indoubt between DRDA requesters and DRDA servers. If APPC commit protocols are being used, lists indoubt transactions between partner LUs. If DRDA commit protocols are being used, lists indoubt transactions between DRDA syncpoint managers.

### Authorization

*sysadm*

### Required Connection

Instance

### Command Syntax

```
►►──LIST DRDA INDOUBT TRANSACTIONS──┬────────────────┬──────────────────────────►◄
                                    └─WITH PROMPTING─┘
```

### Command Parameters

**WITH PROMPTING**
Indicates that indoubt transactions are to be processed. If this parameter is specified, an interactive dialog mode is initiated, permitting the user to commit or roll back indoubt transactions. If this parameter is not specified, indoubt transactions are written to the standard output device, and the interactive dialog mode is not initiated.

**Note:** A forget option is not supported. Once the indoubt transaction is committed or rolled back, the transaction is automatically forgotten.

Interactive dialog mode permits the user to:

- List all indoubt transactions (enter `l`)
- List indoubt transaction number *x* (enter `l`, followed by a valid transaction number)
- Quit (enter `q`)
- Commit transaction number *x* (enter `c`, followed by a valid transaction number)
- Roll back transaction number *x* (enter `r`, followed by a valid transaction number).

**Note:** A blank space must separate the command letter from its argument.

Before a transaction is committed or rolled back, the transaction data is displayed, and the user is asked to confirm the action.

## Usage Notes

DRDA indoubt transactions occur when communication is lost between coordinators and participants in distributed units of work. A distributed unit of work lets a user or application read and update data at multiple locations within a single unit of work. Such work requires a two-phase commit.

The first phase requests all the participants to prepare for a commit. The second phase commits or rolls back the transactions. If a coordinator or participant becomes unavailable after the first phase, the distributed transactions are indoubt.

Before issuing the LIST DRDA INDOUBT TRANSACTIONS command, the application process must be connected to the DB2 Syncpoint Manager (SPM) instance. Use the *spm_name* database manager configuration parameter as the *dbalias* on the CONNECT statement. For more information about using the CONNECT statement, see the *SQL Reference*.

TCP/IP connections, using the SPM to coordinate commits, use DRDA two-phase commit protocols. APPC connections use LU6.2 two-phase commit protocols.

## LIST DATALINKS MANAGERS

Lists the DB2 Data Links Managers that are registered to a specified database.

### Authorization

None

### Command Syntax

```
►►──LIST DATALINKS MANAGERS FOR──┬─DATABASE─┬──dbname──────────────────────────►◄
                                 └─DB───────┘
```

### Command Parameters

**dbname**
Specifies a database name.

### See Also

"ADD DATALINKS MANAGER" on page 118.

## LIST HISTORY

Lists entries in the history file. The history file contains a record of recovery and administrative events. Recovery events include full database and table space level backup, restore, and roll forward. Additional logged events include alter table space, run statistics, reorganize table, drop table, and load.

### Authorization

None

### Required Connection

Instance. An explicit attachment is not required. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command.

### Command Syntax

```
►►──LIST HISTORY──┬───────────────────┬──┬─ALL─────────────────────────────┬──────────►
                  ├─BACKUP────────────┤  ├─SINCE──timestamp────────────────┤
                  ├─ROLLFORWARD───────┤  └─CONTAINING──┬─schema.object_name─┬─┘
                  ├─RUNSTATS──────────┤                └─object_name────────┘
                  ├─REORG─────────────┤
                  ├─ALTER TABLESPACE──┤
                  ├─DROPPED TABLE─────┤
                  └─LOAD──────────────┘

►─FOR──┬──────────┬──database-alias──────────────────────────────────────────────►◄
       ├─DATABASE─┤
       └─DB───────┘
```

### Command Parameters

**HISTORY**
: Lists all recovery and administration events which are currently logged in the history file.

**BACKUP**
: Lists backup and restore operations.

**ROLLFORWARD**
: Lists roll forward operations.

**RUNSTATS**
: Lists RUNSTATS operations.

**REORG**
: Lists table reorganization operations.

**ALTER TABLESPACE**
Lists ALTER TABLESPACE operations.

**DROPPED TABLE**
Lists dropped tables.

**LOAD**
Lists load operations.

**ALL** Lists all entries of the specified type in the history file.

**SINCE timestamp**
A complete time stamp (format *yyyymmddhhnnss*), or an initial prefix (minimum *yyyy*) can be specified. All entries with time stamps equal to or less than the time stamp provided are listed.

**CONTAINING schema.object_name**
This qualified name uniquely identifies a table.

**CONTAINING object_name**
This unqualified name uniquely identifies a table space.

**FOR DATABASE database-alias**
Used to identify the database whose recovery history file is to be listed.

## Examples

```
db2 list history since 19980201 for sample
db2 list history backup containing userspace1 for sample
db2 list history dropped table all for db sample
```

## Usage Notes

The report generated by this command contains the following symbols:

- Device
  - A - ADSM
  - C - Client
  - D - Disk
  - K - Diskette
  - L - Local
  - O - Other
  - P - Pipe
  - S - Server
  - T - Tape
  - U - User Exit
- Object
  - D - Database

  - P - Tablespace
  - T - Table
- Operation
  - B - Backup
  - C - Copy
  - D - Dropped table
  - F - Roll forward
  - G - Reorganize table
  - L - Load
  - Q - Quiesce
  - R - Restore
  - S - Run statistics
  - T - Alter table space
  - U - (for future use)
- Type
  - C - Alter Tablespace (add containers)
  - E - End of log
  - F - Offline
  - I - Insert(LOAD)
  - N - Online
  - P - Point in time
  - R - Alter Tablespace (rebalance)
  - S - Quiesce Share
  - U - Quiesce Update
  - X - Quiesce Exclusive
  - Z - Quiesce Reset
- Entry Status
  - D - Deleted (for future use)
  - E - Expired
  - I - Inactive
  - N - Not yet committed
  - Y - Committed or active

## LIST INDOUBT TRANSACTIONS

Provides a list of transactions that are indoubt. The user can interactively commit, roll back, or forget the indoubt transactions.

The two-phase commit protocol comprises:

1. The PREPARE phase, in which the resource manager writes the log pages to disk, so that it can respond to either a COMMIT or a ROLLBACK primitive
2. The COMMIT (or ROLLBACK) phase, in which the transaction is actually committed or rolled back.

An indoubt transaction is one which has been prepared, but not yet committed or rolled back.

### Scope

This command returns a list of indoubt transactions on the executed node.

### Authorization

*dbadm*

### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

### Command Syntax

```
►►──LIST INDOUBT TRANSACTIONS──┬────────────────┬──────────────────────────►◄
                               └─WITH PROMPTING─┘
```

### Command Parameters

**WITH PROMPTING**
Indicates that indoubt transactions are to be processed. If this parameter is specified, an interactive dialog mode is initiated, permitting the user to commit, roll back, or forget indoubt transactions. If this parameter is not specified, indoubt transactions are written to the standard output device, and the interactive dialog mode is not initiated.

Interactive dialog mode permits the user to:

- List all indoubt transactions (enter `l`)

- List indoubt transaction number *x* (enter `l`, followed by a valid transaction number)
- Quit (enter `q`)
- Commit transaction number *x* (enter `c`, followed by a valid transaction number)
- Roll back transaction number *x* (enter `r`, followed by a valid transaction number)
- Forget transaction number *x* (enter `f`, followed by a valid transaction number).

**Note:** A blank space must separate the command letter from its argument.

Before a transaction is committed, rolled back, or forgotten, the transaction data is displayed, and the user is asked to confirm the action.

## Examples

The following is sample dialog generated by LIST INDOUBT TRANSACTIONS:

```
In-doubt Transactions for Database SAMPLE

1.   originator: XA
     appl_id: *LOCAL.DB2.95051815165159       sequence_no: 0001  status: i
timestamp: 05-18-1997 16:51:59 auth_id: SMITH  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F93DD A92F8C4FF3000000
0000BD

2.   originator: XA
     appl_id: *LOCAL.DATABASE.950407161043  sequence_no: 0002  status: i
timestamp: 04-07-1997 16:10:43  auth_id: JONES  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
0000C1
   .
   .
   .

Enter in-doubt transaction command or 'q' to quit.
e.g. 'c 1' heuristically commits transaction 1.
c/r/f/l/q: c 1

1.   originator: XA
     appl_id: *LOCAL.DB2.95051815165159       sequence_no: 0001  status: i
timestamp: 05-18-1997 16:51:59 auth_id: SMITH  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F93DD A92F8C4FF3000000
0000BD
```

```
Do you want to heuristically commit this in-doubt transaction ? (y/n) y

DB20000I "COMMIT INDOUBT TRANSACTION" completed successfully

c/r/f/l/q: c 5

DB20030E "5" is not a valid in-doubt transaction number.

c/r/f/l/q: l

In-doubt Transactions for Database SAMPLE

1.   originator: XA
     appl_id: *LOCAL.DB2.95051815165159       sequence_no: 0001  status: c
timestamp: 05-18-1997 16:51:59 auth_id: SMITH  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F93DD A92F8C4FF3000000
0000BD

2.   originator: XA
     appl_id: *LOCAL.DATABASE.950407161043  sequence_no: 0002  status: i
timestamp: 04-07-1997 16:10:43  auth_id: JONES  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
0000C1
   .
   .
   .
c/r/f/l/q: r 2

2.   originator: XA
     appl_id: *LOCAL.DATABASE.950407161043  sequence_no: 0002  status: i
timestamp: 04-07-1997 16:10:43  auth_id: JONES  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
0000C1

Do you want to heuristically rollback this in-doubt transaction ? (y/n) y

DB20000I "ROLLBACK INDOUBT TRANSACTION" completed successfully

c/r/f/l/q: l 2

2.   originator: XA
     appl_id: *LOCAL.DATABASE.950407161043  sequence_no: 0002  status: r
timestamp: 04-07-1997 16:10:43  auth_id: JONES  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
0000C1

c/r/f/l/q: f 2

2.   originator: XA
     appl_id: *LOCAL.DATABASE.950407161043  sequence_no: 0002  status: r
timestamp: 04-07-1997 16:10:43  auth_id: JONES  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
0000C1

Do you want to forget this in-doubt transaction ? (y/n) y
```

```
DB20000I "FORGET INDOUBT TRANSACTION" completed successfully

c/r/f/l/q: l 2

2.   originator: XA
     appl_id: *LOCAL.DATABASE.950407161043  sequence_no: 0002  status: f
timestamp: 04-07-1997 16:10:43  auth_id: JONES  log_full: n type: RM
xid: 53514C2000000017 00000000544D4442 00000000002F95FE B62F8C4FF3000000
0000C1

c/r/f/l/q: q
```

## Usage Notes

An indoubt transaction is a global transaction that was left in an indoubt state. This occurs when either the Transaction Manager (TM) or at least one Resource Manager (RM) becomes unavailable after successfully completing the first phase (that is, the PREPARE phase) of the two-phase commit protocol. The RMs do not know whether to commit or to roll back their branch of the transaction until the TM can consolidate its own log with the indoubt status information from the RMs when they again become available. For more information, see the chapter on using DB2 with an XA-compliant Transaction Manager in the *Administration Guide*. An indoubt transaction can also exist in an MPP environment. For more information, see the section on transaction failure recovery on failed database partition servers (in the *Administration Guide*).

If LIST INDOUBT TRANSACTIONS is issued against the currently connected database, the command returns the information on indoubt transactions in that database.

Only transactions whose status is indoubt (i), or missing commit acknowledgement (m), can be committed.

Only transactions whose status is indoubt (i) or ended (e) can be rolled back.

Only transactions whose status is committed (c) or rolled back (r) can be forgotten.

**Note:** In the commit phase of a two-phase commit, the coordinator node waits for commit acknowledgements. If one or more nodes do not reply (for example, because of node failure), the transaction is placed in missing commit acknowledgement state.

Indoubt transaction information is valid only at the time that the command is issued. Once in interactive dialog mode, transaction status may change because of external activities. If this happens, and an attempt is made to process an indoubt transaction which is no longer in an appropriate state, an error message is displayed.

After this type of error occurs, the user should quit (q) the interactive dialog and reissue the LIST INDOUBT TRANSACTIONS WITH PROMPTING command to refresh the information shown.

For more information, see the *Administration Guide*.

## LIST NODE DIRECTORY

Lists the contents of the node directory.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──LIST──┬───────┬──NODE DIRECTORY──┬─────────────┬──────────────────────────────►◄
          └─ADMIN─┘                  └─SHOW DETAIL─┘
```

### Command Parameters

**ADMIN**
: Specifies administration server nodes.

**SHOW DETAIL**
: Specifies that the output should include the following information:

- Remote instance name
- System
- Operating system type

### Examples

The following is sample output from LIST NODE DIRECTORY:

```
 Node Directory

 Number of entries in the directory = 6

Node 1 entry:

 Node name                      = DB2APPC1
 Comment                        = A remote APPC node
 Protocol                       = APPC
 Symbolic destination name      = db2inst1
 Security type                  = PROGRAM
Node 2 entry:

 Node name                      = DB2IPX1
 Comment                        = A remote IPX/SPX node
 Protocol                       = IPXSPX
 File server name               = netwsrv
 Bindery object name            = db2inst1
```

```
Node 3 entry:

 Node name                      = DB2IPX2
 Comment                        = IPX/SPX node using direct addr
 Protocol                       = IPXSPX
 File server name               = *
 Bindery object name            = 09212700.400011527745.879E

Node 4 entry:

 Node name                      = DB2NETB1
 Comment                        = A remote NetBIOS node
 Protocol                       = NETBIOS
 Adapter number                 = 0
 Server NNAME                   = DB2INST1

Node 5 entry:

 Node name                      = DB2TCP1
 Comment                        = A remote TCP/IP node
 Protocol                       = TCPIP
 Hostname                       = tcphost
 Service name                   = db2inst1

Node 6 entry:

 Node name                      = DB2TCP2
 Comment                        = TCP/IP node using IP address
 Protocol                       = TCPIP
 Hostname                       = 9.21.15.235
 Service name                   = db2inst2
```

The common fields are identified as follows:

**Node name**
The name of the remote node. This corresponds to the name entered for the *nodename* parameter when the node was cataloged.

**Comment**
A comment associated with the node, entered when the node was cataloged. To change a comment in the node directory, uncatalog the node, and then catalog it again with the new comment.

**Protocol**
The communications protocol cataloged for the node.

**Note:** For information about fields associated with a specific node type, see the applicable CATALOG...NODE command.

## Usage Notes

A node directory is created and maintained on each database client. It contains an entry for each remote workstation having databases that the client can access. The DB2 client uses the communication end point information in the node directory whenever a database connection or instance attachment is requested.

The database manager creates a node entry and adds it to the node directory each time it processes a CATALOG...NODE command. The entries can vary, depending on the communications protocol being used by the node.

The node directory can contain entries for the following types of nodes:
- APPC
- APPCLU
- APPN
- IPX/SPX
- Local
- Named pipe
- NetBIOS
- TCP/IP.

## See Also

"CATALOG APPC NODE" on page 145

"CATALOG APPCLU NODE" on page 148

"CATALOG APPN NODE" on page 150

"CATALOG IPX/SPX NODE" on page 162

"CATALOG LOCAL NODE" on page 171

"CATALOG NAMED PIPE NODE" on page 173

"CATALOG NETBIOS NODE" on page 175

"CATALOG TCP/IP NODE" on page 179.

## LIST NODEGROUPS

Lists all nodegroups associated with the current database.

### Scope

This command can be issued from any node that is listed in `$HOME/sqllib/db2nodes.cfg`. It returns the same information from any of these nodes.

### Authorization

For the system catalogs `SYSCAT.NODEGROUPS` and `SYSCAT.NODEGROUPDEF`, one of the following is required:

- *sysadm* or *dbadm* authority
- CONTROL privilege
- SELECT privilege.

### Required Connection

Database

### Command Syntax

```
►►──LIST NODEGROUPS──┬─────────────┬──────────────────────────────────►◄
                     └─SHOW DETAIL─┘
```

### Command Parameters

**SHOW DETAIL**
Specifies that the output should include the following information:
- Partitioning map ID
- Node number
- In-use flag

### Examples

Following is sample output from the LIST NODEGROUPS command:

```
NODEGROUP NAME
----------------------------
IBMCATGROUP
IBMDEFAULTGROUP
IBMTEMPGROUP

  3 record(s) selected.
```

Following is sample output from the LIST NODEGROUPS SHOW DETAIL command:

```
NODEGROUP NAME              PMAP_ID NODE NUMBER          IN_USE
--------------------------- ------- -------------------- ------
IBMCATGROUP                       0                    0 Y
IBMDEFAULTGROUP                   1                    0 Y

  2 record(s) selected.
```

The fields are identified as follows:

**NODEGROUP NAME**
The name of the nodegroup. The name is repeated for each node in the nodegroup.

**PMAP_ID**
The ID of the partitioning map. The ID is repeated for each node in the nodegroup.

**NODE NUMBER**
The number of the node.

**IN_USE**
One of four values:

**Y** The node is being used by the nodegroup.

**D** The node is going to be dropped from the nodegroup as a result of a REDISTRIBUTE NODEGROUP operation. When the operation completes, the node will not be included in reports from LIST NODEGROUPS.

**A** The node has been added to the nodegroup but is not yet added to the partitioning map. The containers for the table spaces in the nodegroup have been added on this node. The value is changed to `Y` when the REDISTRIBUTE NODEGROUP operation completes successfully.

**T** The node has been added to the nodegroup, but is not yet added to the partitioning map. The containers for the table spaces in the nodegroup have not been added on this node. Table space containers must be added on the new node for each table space in the node group. The value is changed to `A` when containers have successfully been added.

For more information, see the *SQL Reference*.

## See Also

"REDISTRIBUTE NODEGROUP" on page 409.

## LIST NODES

Lists all nodes associated with the current database.

### Scope

This command can be issued from any node that is listed in `$HOME/sqllib/db2nodes.cfg`. It returns the same information from any of these nodes.

### Authorization

None

### Required Connection

Database

### Command Syntax

```
►►──LIST NODES─────────────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

Following is sample output from the LIST NODES command:

```
NODE NUMBER
----------------------
                     0
                     2
                     5
                     7
                     9

  5 record(s) selected.
```

### See Also

"REDISTRIBUTE NODEGROUP" on page 409.

## LIST ODBC DATA SOURCES

Lists all available user or system ODBC data sources.

A *data source*, in ODBC (Open Database Connectivity) terminology, is a user-defined name for a specific database or file system. That name is used to access the database or file system through ODBC APIs. On Windows NT and Windows 95, either user or system data sources can be cataloged. A user data source is only visible to the user who cataloged it, whereas a system data source is visible to and can be used by all other users.

This command is available on Windows NT, Windows 95, and Windows 3.1 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
            ┌─USER───┐
►►──LIST────┼────────┼──ODBC DATA SOURCES───────────────────────────────────►◄
            └─SYSTEM─┘
```

### Command Parameters

**USER** List only user ODBC data sources. This is the default if no keyword is specified.

**SYSTEM**
List only system ODBC data sources.

### Examples

The following is sample output from the LIST ODBC DATA SOURCES command:

```
               User ODBC Data Sources

Data source name                 Description
-------------------------------- ----------------------------------------
SAMPLE                           IBM DB2 ODBC DRIVER
```

## LIST ODBC DATA SOURCES

### See Also

"CATALOG ODBC DATA SOURCE" on page 178

"UNCATALOG ODBC DATA SOURCE" on page 493.

## LIST PACKAGES/TABLES

Lists packages or tables associated with the current database.

### Authorization

For the system catalogs `SYSCAT.PACKAGES` (LIST PACKAGES) and `SYSCAT.TABLES` (LIST TABLES), one of the following is required:

- *sysadm* or *dbadm* authority
- CONTROL privilege
- SELECT privilege.

### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

### Command Syntax

```
>>──LIST──┬─PACKAGES─┬──┬─────────────────────────────────┬──┬─────────────┬──────────><
          └─TABLES───┘  │       ┌─USER──────────────────┐ │  └─SHOW DETAIL─┘
                        └─FOR───┼─ALL───────────────────┼─┘
                                ├─SCHEMA──schema-name───┤
                                └─SYSTEM────────────────┘
```

### Command Parameters

**FOR** If the FOR clause is not specified, the packages or tables for USER are listed.

**ALL** Lists all packages or tables in the database.

**SCHEMA**
Lists all packages or tables in the database for the specified schema only.

**SYSTEM**
Lists all system packages or tables in the database.

**USER** Lists all user packages or tables in the database for the current user.

**SHOW DETAIL**
Displays the full table name (valid for the LIST TABLES command only). If this option is not specified, the name is truncated to 30 characters, and the ″>″ symbol in the thirty-first column represents the truncated portion of the table name.

## Examples

The following is sample output from LIST PACKAGES:

```
                      Bound    Total                                Isolation
Package    Schema     by       sections      Valid    Format   level     Blocking
---------- --------- -------- ------------- -------- -------- --------- --------
P1         SMITH     SMITH                1 Yes      0        CS        U

  1 record(s) selected.
```

The following is sample output from LIST TABLES:

```
Table/View         Schema           Type       Creation time
------------------ ---------------- ---------- ----------------------------
DEPARTMENT         SMITH            T          1997-02-19-13.32.25.971890
EMP_ACT            SMITH            T          1997-02-19-13.32.27.851115
EMP_PHOTO          SMITH            T          1997-02-19-13.32.29.953624
EMP_RESUME         SMITH            T          1997-02-19-13.32.37.837433
EMPLOYEE           SMITH            T          1997-02-19-13.32.26.348245
ORG                SMITH            T          1997-02-19-13.32.24.478021
PROJECT            SMITH            T          1997-02-19-13.32.29.300304
SALES              SMITH            T          1997-02-19-13.32.42.973739
STAFF              SMITH            T          1997-02-19-13.32.25.156337

  9 record(s) selected.
```

## Usage Notes

LIST PACKAGES and LIST TABLES commands are available to provide a quick Version 1 interface to the system tables. However, Version 2 system tables offer a greater granularity of information, and should be used whenever possible.

The following Version 2 SELECT statements return similar information. They can be expanded to select the additional information that Version 2 provides.

```
   select tabname, tabschema, type, create_time
   from syscat.tables
   order by tabschema, tabname;

   select pkgname, pkgschema, boundby, total_sect,
      valid, format, isolation, blocking
   from syscat.packages
   order by pkgschema, pkgname;

   select tabname, tabschema, type, create_time
   from syscat.tables
   where tabschema = 'SYSCAT'
   order by tabschema, tabname;
```

```
select pkgname, pkgschema, boundby, total_sect,
   valid, format, isolation, blocking
from syscat.packages
where pkgschema = 'NULLID'
order by pkgschema, pkgname;

select tabname, tabschema, type, create_time
from syscat.tables
where tabschema = USER
order by tabschema, tabname;

select pkgname, pkgschema, boundby, total_sect,
   valid, format, isolation, blocking
from syscat.packages
where pkgschema = USER
order by pkgschema, pkgname;
```

## LIST TABLESPACE CONTAINERS

Lists containers for the specified table space.

### Scope

This command returns information only for the node on which it is executed.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

### Required Connection

Database

### Command Syntax

```
►►──LIST TABLESPACE CONTAINERS FOR──tablespace-id──┬─────────────┬──────────────────►◄
                                                   └─SHOW DETAIL─┘
```

### Command Parameters

**FOR tablespace-id**
An integer that uniquely represents a table space used by the current database. To get a list of all the table spaces used by the current database, use "LIST TABLESPACES" on page 332.

**SHOW DETAIL**
If this option is not specified, only the following basic information about each container is provided:
- Container ID
- Name
- Type (file, disk, or path).

If this option is specified, the following additional information about each container is provided:
- Total number of pages
- Number of useable pages
- Accessible (yes or no).

## Examples

The following is sample output from LIST TABLESPACE CONTAINERS:

```
            Tablespace Containers for Tablespace 0

 Container ID                         = 0
 Name                                 = /home/smith/smith/NODE0000/
                                        SQL00001/SQLT0000.0
 Type                                 = Path
```

The following is sample output from LIST TABLESPACE CONTAINERS with SHOW DETAIL specified:

```
            Tablespace Containers for Tablespace 0

 Container ID                         = 0
 Name                                 = /home/smith/smith/NODE0000/
                                        SQL00001/SQLT0000.0
 Type                                 = Path
 Total pages                          = 895
 Useable pages                        = 895
 Accessible                           = Yes
```

## See Also

"LIST TABLESPACES" on page 332.

## LIST TABLESPACES

Lists table spaces for the current database.

### Scope

This command returns information only for the node on which it is executed.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

### Required Connection

Database

### Command Syntax

```
►►──LIST TABLESPACES──┬─────────────┬──────────────────────────────────────►◄
                      └─SHOW DETAIL─┘
```

### Command Parameters

**SHOW DETAIL**
If this option is not specified, only the following basic information about each table space is provided:
- Table space ID
- Name
- Type (system managed space or database managed space)
- Contents (any data, long data only, or temporary data)
- State, a hexadecimal value indicating the current table space state. The externally visible state of a table space is composed of the hexadecimal sum of certain state values. For example, if the state is ″quiesced: EXCLUSIVE″ and ″Load pending″, the value is `0x0004` + `0x0008`, which is `0x000c`. "db2tbst - Get Tablespace State" on page 83 can be used to obtain the table space state associated with a given hexadecimal value. Following are the bit definitions listed in `sqlutil.h`:

```
0x0        Normal
0x1        Quiesced: SHARE
0x2        Quiesced: UPDATE
```

```
0x4         Quiesced: EXCLUSIVE
0x8         Load pending
0x10        Delete pending
0x20        Backup pending
0x40        Roll forward in progress
0x80        Roll forward pending
0x100       Restore pending
0x100       Recovery pending (not used)
0x200       Disable pending
0x400       Reorg in progress
0x800       Backup in progress
0x1000      Storage must be defined
0x2000      Restore in progress
0x4000      Offline and not accessible
0x8000      Drop pending
0x2000000   Storage may be defined
0x4000000   StorDef is in 'final' state
0x8000000   StorDef was changed prior to rollforward
0x10000000  DMS rebalancer is active
0x20000000  TBS deletion in progress
0x40000000  TBS creation in progress
0x8         For service use only
```

If this option is specified, the following additional information about each table space is provided:

- Total number of pages
- Number of useable pages
- Number of used pages
- Number of free pages
- High water mark (in pages)
- Page size (in bytes)
- Extent size (in pages)
- Prefetch size (in pages)
- Number of containers
- Minimum recovery time (displayed only if not zero)
- State change table space ID (displayed only if the table space state is ″load pending″ or ″delete pending″)
- State change object ID (displayed only if the table space state is ″load pending″ or ″delete pending″)
- Number of quiescers (displayed only if the table space state is ″quiesced: SHARE″, ″quiesced: UPDATE″, or ″quiesced: EXCLUSIVE″)
- Table space ID and object ID for each quiescer (displayed only if the number of quiescers is greater than zero).

## Examples

The following are two sample outputs from LIST TABLESPACES SHOW DETAIL.

```
          Tablespaces for Current Database
 Tablespace ID                        = 0
 Name                                 = SYSCATSPACE
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 895
 Useable pages                        = 895
 Used pages                           = 895
 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
 Tablespace ID                        = 1
 Name                                 = TEMPSPACE1
 Type                                 = System managed space
 Contents                             = Temporary data
 State                                = 0x0000
   Detailed explanation:
     Normal
 Total pages                          = 1
 Useable pages                        = 1
 Used pages                           = 1
 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
 Tablespace ID                        = 2
 Name                                 = USERSPACE1
 Type                                 = System managed space
 Contents                             = Any data
 State                                = 0x000c
   Detailed explanation:
     Quiesced: EXCLUSIVE
     Load pending
 Total pages                          = 337
 Useable pages                        = 337
 Used pages                           = 337
 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
DB21011I  In a partitioned database server environment, only the table spaces
on the current node are listed.

           Tablespaces for Current Database
 Tablespace ID                       = 0
 Name                                = SYSCATSPACE
 Type                                = System managed space
 Contents                            = Any data
 State                               = 0x0000
   Detailed explanation:
     Normal
 Total pages                         = 1200
 Useable pages                       = 1200
 Used pages                          = 1200
 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
 Tablespace ID                       = 1
 Name                                = TEMPSPACE1
 Type                                = System managed space
 Contents                            = Temporary data
 State                               = 0x0000
   Detailed explanation:
     Normal
 Total pages                         = 1
 Useable pages                       = 1
 Used pages                          = 1
 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
Tablespace ID                        = 2
 Name                                = USERSPACE1
 Type                                = System managed space
 Contents                            = Any data
 State                               = 0x0000
   Detailed explanation:
     Normal
 Total pages                         = 1
 Useable pages                       = 1
 Used pages                          = 1
 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
Tablespace ID                          = 3
 Name                                  = DMS8K
 Type                                  = Database managed space
 Contents                              = Any data
 State                                 = 0x0000
   Detailed explanation:
     Normal
 Total pages                           = 2000
 Useable pages                         = 1952
 Used pages                            = 96
 Free pages                            = 1856
 High water mark (pages)               = 96
 Page size (bytes)                     = 8192
 Extent size (pages)                   = 32
 Prefetch size (pages)                 = 32
 Number of containers                  = 2
Tablespace ID                          = 4
 Name                                  = TEMP8K
 Type                                  = System managed space
 Contents                              = Temporary data
 State                                 = 0x0000
   Detailed explanation:
     Normal
 Total pages                           = 1
 Useable pages                         = 1
 Used pages                            = 1
 Free pages                            = Not applicable
 High water mark (pages)               = Not applicable
 Page size (bytes)                     = 8192
 Extent size (pages)                   = 32
 Prefetch size (pages)                 = 32
 Number of containers                  = 1
DB21011I  In a partitioned database server environment, only the table spaces
on the current node are listed.
```

## Usage Notes

In a multi-node environment, this command does not return all the table spaces in the database. To obtain a list of all the table spaces, query SYSCAT.SYSTABLESPACES.

During a table space rebalance, the number of useable pages will include pages for the newly added container, but these new pages will not be reflected in the number of free pages until the rebalance is complete. When a table space rebalance is *not* taking place, the number of used pages plus the number of free pages will equal the number of useable pages.

For detailed information about tables spaces, see the *Administration Guide*.

## See Also

"db2tbst - Get Tablespace State" on page 83

"LIST TABLESPACE CONTAINERS" on page 330.

## LOAD

Loads data from files, tapes, or named pipes into a DB2 table. Tape is not supported on OS/2. The load utility does not support loading data at the hierarchy level.

### Scope

This command affects only the partition to which a direct connection exists; the load utility operates on a single database partition only.

### Authorization

One of the following:

- *sysadm*
- *dbadm*
- load authority on the database and
  - INSERT privilege on the table when the load utility is invoked in APPEND, TERMINATE, or RESTART mode
  - INSERT and DELETE privilege on the table when the load utility is invoked in REPLACE mode.

**Note:** Since all load processes (and all DB2 server processes, in general), are owned by the instance owner, and all of these processes use the identification of the instance owner to access needed files, the instance owner must have read access to input data files. These input data files must be readable by the instance owner, regardless of who invokes the command.

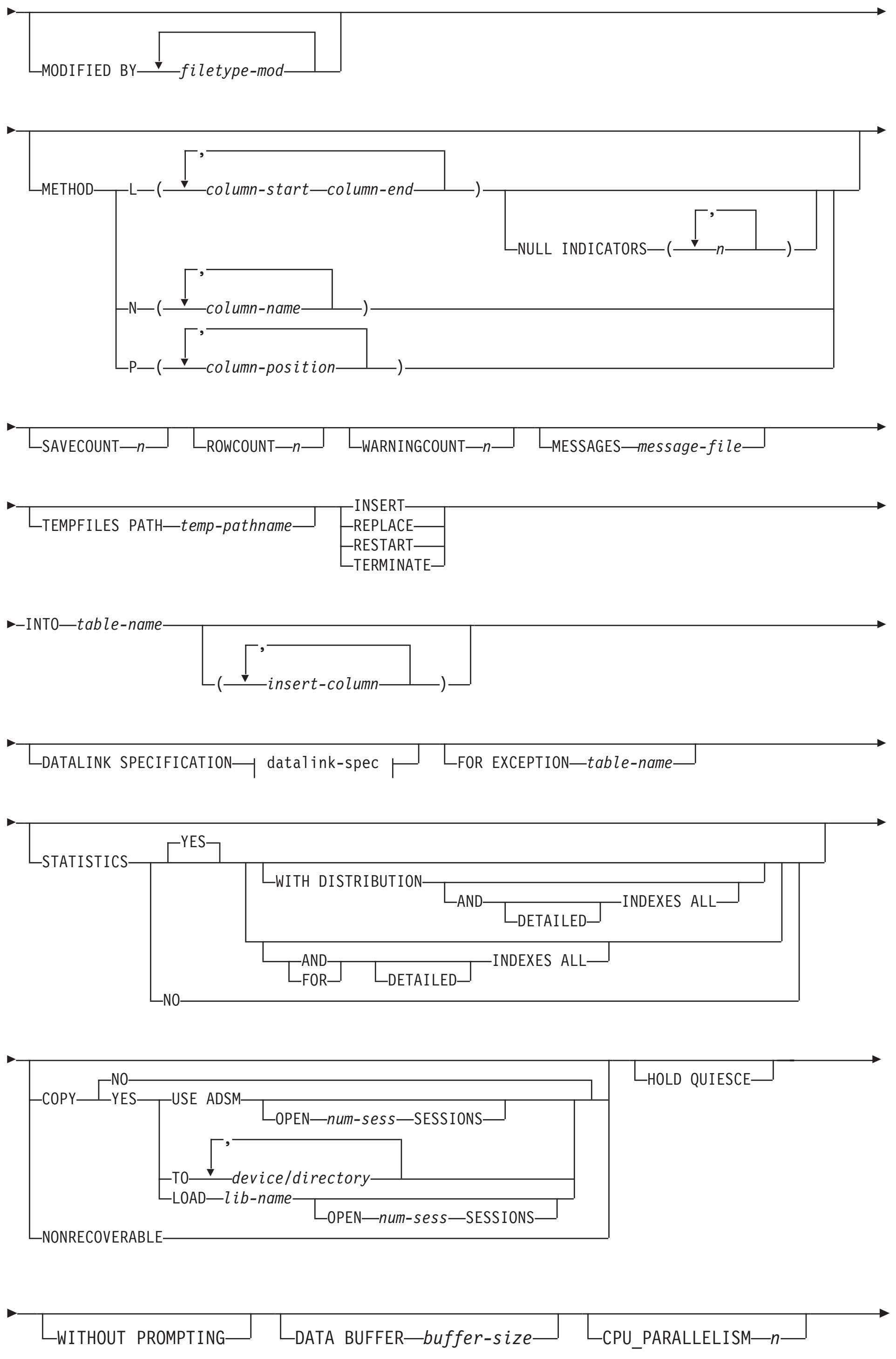
### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

Instance. An explicit attachment is not required. If a connection to the database has been established, an implicit attachment to the local instance is attempted.

### Command Syntax

```
              ┌─,──────────────┐
              ▼                │
►►──LOAD FROM────┬─filename─┬──┴──OF──filetype──┬─────────────────────────────────┬──────────────►
                 ├─pipename─┤                   │              ┌─,──────────┐    │
                 └─device───┘                   │              ▼            │    │
                                                └─LOBS FROM────lob-path─────┴────┘
```


MODIFIED BY
filetype-mod
METHOD
L
(
,
column-start
column-end
)
NULL INDICATORS
(
,
n
)
N
(
,
column-name
)
P
(
,
column-position
)
SAVECOUNT
n
ROWCOUNT
n
WARNINGCOUNT
n
MESSAGES
message-file
TEMPFILES PATH
temp-pathname
INSERT
REPLACE
RESTART
TERMINATE
INTO
table-name
(
,
insert-column
)
DATALINK SPECIFICATION
datalink-spec
FOR EXCEPTION
table-name
STATISTICS
YES
WITH DISTRIBUTION
AND
DETAILED
INDEXES ALL
AND
FOR
DETAILED
INDEXES ALL
NO
COPY
NO
YES
USE ADSM
OPEN
num-sess
SESSIONS
TO
,
device/directory
LOAD
lib-name
OPEN
num-sess
SESSIONS
NONRECOVERABLE
HOLD QUIESCE
WITHOUT PROMPTING
DATA BUFFER
buffer-size
CPU_PARALLELISM
n

```
►─┬─────────────────────┬──┬──────────────────────────────────────┬──────────────────────────►◄
  └─DISK_PARALLELISM──n─┘  └─INDEXING MODE──┬─AUTOSELECT──┬─┘
                                            ├─REBUILD─────┤
                                            ├─INCREMENTAL─┤
                                            └─DEFERRED────┘
```

**datalink-spec:**

```
     ┌─,──────────────────────────────────────────────────────────────────────────────────────────┐
     ▼                                                                                            │
├──(──┬─────────────────┬──┬───────────────────────────────────┬──┬─────────────────────────────┬──)──┴──┤
      └─DL_LINKTYPE URL─┘  ├─DL_URL_REPLACE_PREFIX──"prefix"───┤  └─DL_URL_SUFFIX──"suffix"────┘
                           └─DL_URL_DEFAULT_PREFIX──"prefix"───┘
```

## Command Parameters

**COPY NO**
Specifies that the table space in which the table resides will be placed in backup pending state if forward recovery is enabled (that is, *logretain* or *userexit* is on). The data will not be accessible until a table space backup or a full database backup is made.

**COPY YES**
Specifies that a copy of the loaded data will be saved. This option is invalid if forward recovery is disabled (both *logretain* and *userexit* are off). The option is not supported for tables with DATALINK columns.

**USE ADSM**
Specifies that the copy will be stored using ADSTAR Distributed Storage Manager (ADSM).

**OPEN num-sess SESSIONS**
The number of I/O sessions to be used with ADSM or the vendor product. The default value is 1.

**TO device/directory**
Specifies the device or directory on which the copy image will be created. Tape is not supported on OS/2; copy to tapes is not supported for DB2 servers running on SCO UnixWare 7.

**LOAD lib-name**
The name of the shared library (DLL on OS/2 or the Windows operating system) containing the vendor backup and restore I/O functions to be used. It may contain the full path. If the full path is not given, it will default to the path where the user exit programs reside.

**CPU_PARALLELISM n**
Specifies the number of processes or threads that the load utility will spawn for parsing, converting, and formatting records when building

table objects. This parameter is designed to exploit intra-partition parallelism. It is particularly useful when loading presorted data, because record order in the source data is preserved. If the value of this parameter is zero, or has not been specified, the load utility uses an intelligent default value at run time.

**Notes:**

1. If this parameter is used with tables containing either LOB or LONG VARCHAR fields, its value becomes one, regardless of the number of system CPUs or the value specified by the user.
2. Specifying a small value for the SAVECOUNT parameter causes the loader to perform many more I/O operations to flush both data and table metadata. When CPU_PARALLELISM is greater than one, the flushing operations are asynchronous, permitting the loader to exploit the CPU. When CPU_PARALLELISM is set to one, the loader waits on I/O during consistency points. A load operation with CPU_PARALLELISM set to two, and SAVECOUNT set to 10 000, completes faster than the same operation with CPU_PARALLELISM set to one, even though there is only one CPU.

**DATA BUFFER buffer-size**

Specifies the number of 4KB pages (regardless of the degree of parallelism) to use as buffered space for transferring data within the utility. If the value specified is less than the algorithmic minimum, the minimum required resource is used, and no warning is returned.

This memory is allocated directly from the utility heap, whose size can be modified through the *util_heap_sz* database configuration parameter.

If a value is not specified, an intelligent default is calculated by the utility at run time. The default is based on a percentage of the free space available in the utility heap at the instantiation time of the loader, as well as some characteristics of the table.

**DATALINK SPECIFICATION**

For each DATALINK column, there can be one column specification enclosed by parentheses. Each column specification consists of one or more DL_LINKTYPE, prefix, and a DL_URL_SUFFIX specification. The prefix specification can be either DL_URL_REPLACE_PREFIX or DL_URL_DEFAULT_PREFIX.

There can be as many DATALINK column specifications as the number of DATALINK columns defined in the table. The order of specifications follows the order of DATALINK columns found within the *insert-column* list, or within the table definition (if an *insert-column* list is not specified).

**DISK_PARALLELISM n**
Specifies the number of processes or threads that the load utility will spawn for writing data to the table space containers. If a value is not specified, the utility selects an intelligent default based on the number of table space containers and the characteristics of the table.

**DL_LINKTYPE**
If specified, it should match the LINKTYPE of the column definition. Thus, DL_LINKTYPE URL is acceptable if the column definition specifies LINKTYPE URL.

**DL_URL_DEFAULT_PREFIX** ″**prefix**″
If specified, it should act as the default prefix for all DATALINK values within the same column. In this context, prefix refers to the ″scheme host port″ part of the URL specification.

Examples of prefix are:

```
"http://server"
"file://server"
"file:"
"http://server:80"
```

If no prefix is found in the column data, and a default prefix is specified with DL_URL_DEFAULT_PREFIX, the default prefix is prefixed to the column value (if not NULL).

For example, if DL_URL_DEFAULT_PREFIX specifies the default prefix `"http://toronto"`:

- The column input value ″/x/y/z″ is stored as ″http://toronto/x/y/z″.
- The column input value ″http://coyote/a/b/c″ is stored as ″http://coyote/a/b/c″.
- The column input value NULL is stored as NULL.

**DL_URL_REPLACE_PREFIX** ″**prefix**″
This clause is useful when loading or importing data previously generated by the export utility, if the user wants to globally replace the host name in the data with another host name. If specified, it becomes the prefix for *all* non-NULL column values. If a column value has a prefix, this will replace it. If a column value has no prefix, the prefix specified by DL_URL_REPLACE_PREFIX is prefixed to the column value.

For example, if DL_URL_REPLACE_PREFIX specifies the prefix `"http://toronto"`:

- The column input value ″/x/y/z″ is stored as ″http://toronto/x/y/z″.

- The column input value ″http://coyote/a/b/c″ is stored as ″http://toronto/a/b/c″. Note that ″toronto″ replaces ″coyote″.
- The column input value NULL is stored as NULL.

**DL_URL_SUFFIX ″suffix″**

If specified, it is appended to every non-NULL column value for the column. It is, in fact, appended to the ″path″ component of the data location part of the DATALINK value.

**FOR EXCEPTION table-name**

Specifies the exception table into which rows in error will be copied. Any row that is in violation of a unique index or a primary key index is copied. DATALINK exceptions are also captured in the exception table.

Information that is written to the exception table is *not* written to the dump file (for a description of the `dumpfile` modifier, see Table 7 on page 359). In a partitioned database environment, an exception table must be defined for those nodes on which the loading table is defined. The dump file, on the other hand, contains rows that cannot be loaded because they are invalid or have syntax errors. For more information, see the *Data Movement Utilities Guide and Reference*.

**FROM filename/pipename/device**

Specifies the file, pipe, or device that contains the data being loaded. This file, pipe, or device must reside on the node where the database resides. If several names are specified, they will be processed in sequence. If the last item specified is a tape device, the user is prompted for another tape. Valid response options are:

**c** Continue. Continue using the device that generated the warning message (for example, when a new tape has been mounted).

**d** Device terminate. Stop using the device that generated the warning message (for example, when there are no more tapes).

**t** Terminate. Terminate all devices.

**Notes:**

1. Tape is not supported on OS/2.
2. It is recommended that the fully qualified file name be used. If the server is remote, the fully qualified file name must be used. If the database resides on the same node as the caller, relative paths may be used.
3. Loading data from multiple IXF files is supported if the files are physically separate, but logically one file. It is *not* supported if the files are both logically and physically separate.

4. If, when specifying *pipename* on OS/2, less than the expected amount of data is loaded, clean up system resources (IPL is recommended), and reissue the LOAD command.

**HOLD QUIESCE**
Specifies that the utility should leave the table in quiesced exclusive state after the load operation. To unquiesce the table spaces, issue:

```
db2 quiesce tablespaces for table <tablename> reset
```

**Note:** Ensure that no *phantom quiesces* are created (see “QUIESCE TABLESPACES FOR TABLE” on page 399).

**INDEXING MODE**
Specifies whether the load utility is to rebuild indexes or to extend them incrementally. Valid values are:

**AUTOSELECT**
The load utility will automatically decide between REBUILD or INCREMENTAL mode.

**REBUILD**
All indexes will be rebuilt. The utility must have sufficient resources to sort all index key parts for both old and appended table data.

**INCREMENTAL**
Indexes will be extended with new data. This approach consumes index free space. It only requires enough sort space to append index keys for the inserted records. This method is only supported in cases where the index object is valid and accessible at the start of a load operation (it is, for example, not valid immediately following a load operation in which the DEFERRED mode was specified). If this mode is specified, but not supported due to the state of the index, a warning is returned, and the load operation continues in REBUILD mode. Similarly, if a load restart operation is begun in the load build phase, INCREMENTAL mode is not supported.

Incremental indexing is not supported when all of the following conditions is true:

- The LOAD COPY option is specified (*logretain* or *userexit* is enabled).
- The table resides in a DMS table space.
- The index object resides in a table space that is shared by other table objects belonging to the table being loaded.

To bypass this restriction, it is recommended that indexes be placed in a separate table space.

**DEFERRED**

The load utility will not attempt index creation if this mode is specified. Indexes will be marked as needing a refresh. The first access to such indexes that is unrelated to a load operation may force a rebuild (for more information, see the *Administration Guide*), or indexes may be rebuilt when the database is restarted. This approach requires enough sort space for all key parts for the largest index. The total time subsequently taken for index construction is longer than that required in REBUILD mode. Therefore, when performing multiple load operations with deferred indexing, it is advisable (from a performance viewpoint) to let the last load operation in the sequence perform an index rebuild, rather than allow indexes to be rebuilt at first non-load access.

Deferred indexing is only supported for tables with non-unique indexes, so that duplicate keys inserted during the load phase are not persistent after the load operation.

**INSERT**

One of four modes under which the load utility can execute. Adds the loaded data to the table without changing the existing table data.

**insert-column**

Specifies the table column into which the data is to be inserted.

The load utility cannot parse columns whose names contain one or more spaces. For example,

```
db2 load from delfile1 of del modified by noeofchar noheader
   method P (1, 2, 3, 4, 5, 6, 7, 8, 9)
   insert into table1 (BLOB1, S2, I3, Int 4, I5, I6, DT7, I8, TM9)
```

will fail because of the `Int 4` column. The solution is to enclose such column names with double quotation marks:

```
db2 load from delfile1 of del modified by noeofchar noheader
   method P (1, 2, 3, 4, 5, 6, 7, 8, 9)
   insert into table1 (BLOB1, S2, I3, "Int 4", I5, I6, DT7, I8, TM9)
```

**INTO table-name**

Specifies the database table into which the data is to be loaded. This table cannot be a system table. An alias, or the fully qualified or unqualified table name can be specified. A qualified table name is in the form *schema.tablename*. If an unqualified table name is specified, the table will be qualified with the current authorization ID.

**LOBS FROM lob-path**

The path to the data files containing LOB values to be loaded. The path must end with a slash (/). The names of the LOB data files are stored in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the LOB column. This option is ignored if `lobsinfile` is not specified within the *filetype-mod* string (see Table 7 on page 359).

**MESSAGES message-file**

Specifies the destination for warning and error messages that occur during the load operation. If a message file is not specified, messages are written to standard output. If the complete path to the file is not specified, the load utility uses the current directory and the default drive as the destination. If the name of a file that already exists is specified, the utility appends the information.

**METHOD**

**L** Specifies the start and end column numbers from which to load data.

**Note:** This method can only be used with ASC files, and is the only valid option for that file type.

**N** Specifies the names of the columns in the data file to be loaded. The case of these column names must match the case of the corresponding names in the system catalogs. Each column in the table that is not nullable should be included in this list. Specify only complete subsets of column names (for example, given file columns F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT NULL, C3 INT NOT NULL, and C4 INT, `method N (F1,F2,F3,F4) insert into table_name (C1,C2,C3,C4)` is a valid request, while `method N (F1,F4)` is not valid, since there will be no data to put into C3.

**Note:** This method can only be used with IXF files.

**P** Specifies the numbers of the columns to be loaded. Each column in the table that is not nullable should be included in this list. Specify only complete subsets of column numbers (for example, given file columns F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT NULL, C3 INT NOT NULL, and C4 INT, `method P (1,2,3,4)` is a valid request, while `method P (1,4)` is not valid.

**Note:** This method can only be used with IXF or DEL files, and is the only valid option for the DEL file type.

**MODIFIED BY filetype-mod**

Specifies additional options (see Table 7 on page 359).

**NONRECOVERABLE**

Specifies that the load transaction is to be marked as non-recoverable, and that it will not be possible to recover it by a subsequent roll forward action. The rollforward utility will skip the transaction, and will mark the table into which data was being loaded as "invalid". The utility will also ignore any subsequent transactions against that table. After the roll forward is completed, such a table can only be dropped.

With this option, table spaces are not put in backup pending state following the load operation, and a copy of the loaded data does not have to be made during the load operation.

This option should not be used when DATALINK columns with the FILE LINK CONTROL attribute are present in, or being added to, the table.

**NULL INDICATORS n**

Specifies a column (by number) to be used as a NULL indicator field. If this option is used, a NULL indicator column for each data column must also be specified. A value of zero indicates that the data column is not nullable, and that there will always be data in that column.

A value of `Y` in the NULL indicator column specifies that the column data is NULL. Any character *other than* `Y` in the NULL indicator column specifies that the column data is not NULL, and that column data specified by the METHOD L option will be loaded.

The NULL indicator character can be changed using the MODIFIED BY option (see the description of the `nullindchar` modifier in Table 7 on page 359).

**OF filetype**

Specifies the format of the data in the input file:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format)
- IXF (integrated exchange format, PC version), exported from the same or from another DB2 table.

For more information about file formats, see the "Export/Import/Load Utility File Formats" appendix in the *Data Movement Utilities Guide and Reference.*

**REPLACE**

One of four modes under which the load utility can execute. Deletes all existing data from the table, and inserts the loaded data. The table definition and index definitions are not changed. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

This option is not supported for tables with DATALINK columns.

**RESTART**
: One of four modes under which the load utility can execute. Restarts a previously interrupted load operation. The load operation will automatically continue from the last consistency point in the load, build, or delete phase.

**RESTARTCOUNT**
: Reserved.

**ROWCOUNT n**
: Specifies the number of *n* physical records in the file to be loaded. Allows a user to load only the first *n* rows in a file.

**SAVECOUNT n**
: Specifies that the load utility is to establish consistency points after every *n* rows. This value is converted to a page count, and rounded up to intervals of the extent size. Since a message is issued at each consistency point, this option should be selected if the load operation will be monitored using “LOAD QUERY” on page 368. If the value of *n* is not sufficiently high, the synchronization of activities performed at each consistency point will impact performance.

  The default value is zero, meaning that no consistency points will be established, unless necessary.

**SORT BUFFER buffer-size**
: Reserved.

**STATISTICS NO**
: Specifies that no statistics are to be collected, and that the statistics in the catalogs are not to be altered. This is the default.

**STATISTICS YES**
: Specifies that statistics are to be collected for the table and for any existing indexes. This option is supported only if the load operation is in REPLACE mode.

  **WITH DISTRIBUTION**
  : Specifies that distribution statistics are to be collected.

  **AND INDEXES ALL**
  : Specifies that both table and index statistics are to be collected.

  **FOR INDEXES ALL**
  : Specifies that only index statistics are to be collected.

  **DETAILED**
  : Specifies that extended index statistics are to be collected.

**TEMPFILES PATH temp-pathname**

Specifies the name of the path to be used when creating temporary files during a load operation, and should be fully qualified according to the server node.

Temporary files take up file system space. Sometimes, this space requirement is quite substantial. Following is an estimate of how much file system space should be allocated for all temporary files:

- 4 bytes for each duplicate or rejected row containing DATALINK values
- 136 bytes for each message that the load utility generates
- 15KB overhead if the data file contains long field data or LOBs. This quantity can grow significantly if the INSERT option is specified, and there is a large amount of long field or LOB data already in the table.

For more information about temporary files, see the *Data Movement Utilities Guide and Reference*.

**TERMINATE**

One of four modes under which the load utility can execute. Terminates a previously interrupted load operation, and rolls back the operation to the point in time at which it started, even if consistency points were passed. The states of any table spaces involved in the operation return to normal, and all table objects are made consistent (index objects may be marked as invalid, in which case index rebuild will automatically take place at next access). If the load operation being terminated is a load REPLACE, the table will be truncated to an empty table after the load TERMINATE operation. If the load operation being terminated is a load INSERT, the table will retain all of its original records after the load TERMINATE operation.

If the table spaces in which the table resides are not in load pending state, this option does not affect the state of the table spaces.

The load terminate option will not remove a backup pending state from table spaces.

**USING directory**

Reserved.

**WARNINGCOUNT n**

Stops the load operation after *n* warnings. Set this parameter if no warnings are expected, but verification that the correct file and table are being used is desired. If *n* is zero, or this option is not specified, the load operation will continue regardless of the number of warnings issued. If the load operation is stopped because the threshold of warnings was encountered, another load operation can be started in

RESTART mode. The load operation will automatically continue from the last consistency point. Alternatively, another load operation can be initiated in REPLACE mode, starting at the beginning of the input file.

**WITHOUT PROMPTING**
Specifies that the list of data files contains all the files that are to be loaded, and that the devices or directories listed are sufficient for the entire load operation. If a continuation input file is not found, or the copy targets are filled before the load operation finishes, the load operation will fail, and the table will remain in load pending state.

If this option is not specified, and the tape device encounters an end of tape for the copy image, or the last item listed is a tape device, the user is prompted for a new tape on that device. Tape is not supported on OS/2.

## Examples

**Example 1**

TABLE1 has 5 columns:

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 has 6 elements:

- ELE1 positions 01 to 20
- ELE2 positions 21 to 22
- ELE5 positions 23 to 23
- ELE3 positions 24 to 27
- ELE4 positions 28 to 31
- ELE6 positions 32 to 32
- ELE6 positions 33 to 40

Data Records:

```
1...5....10...15...20...25...30...35...40
Test data 1         XXN 123abcdN
Test data 2 and 3   QQY    wxyzN
Test data 4,5 and 6 WWN6789    Y
```

The following command loads the table from the file:

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
   method L (1 20, 21 22, 24 27, 28 31)
   null indicators (0,0,23,32)
   insert into table1 (col1, col5, col2, col3)
```

**Notes:**

1. The specification of `striptblanks` in the MODIFIED BY parameter forces the truncation of blanks in VARCHAR columns (COL1, for example, which is 11, 17 and 19 bytes long, in rows 1, 2 and 3, respectively).
2. The specification of `reclen=40` in the MODIFIED BY parameter indicates that there is no new-line character at the end of each input record, and that each record is 40 bytes long. The last 8 bytes are not used to load the table.
3. Since COL4 is not provided in the input file, it will be inserted into TABLE1 with its default value (it is defined NOT NULL WITH DEFAULT).
4. Positions 23 and 32 are used to indicate whether COL2 and COL3 of TABLE1 will be loaded NULL for a given row. If there is a `Y` in the column's null indicator position for a given record, the column will be NULL. If there is an `N`, the data values in the column's data positions of the input record (as defined in L(........)) are used as the source of column data for the row. In this example, neither column in row 1 is NULL; COL2 in row 2 is NULL; and COL3 in row 3 is NULL.
5. In this example, the NULL INDICATORS for COL1 and COL5 are specified as 0 (zero), indicating that the data is not nullable.
6. The NULL INDICATOR for a given column can be anywhere in the input record, but the position must be specified, and the `Y` or `N` values must be supplied.

**Example 2 (Loading LOBs from Files)**

TABLE1 has 3 columns:

- COL1 CHAR 4 NOT NULL WITH DEFAULT
- LOB1 LOB
- LOB2 LOB

ASCFILE1 has 3 elements:

- ELE1 positions 01 to 04
- ELE2 positions 06 to 13
- ELE3 positions 15 to 22

The following files reside in either `/u/user1` or `/u/user1/bin`:

- ASCFILE2 has LOB data
- ASCFILE3 has LOB data
- ASCFILE4 has LOB data
- ASCFILE5 has LOB data
- ASCFILE6 has LOB data
- ASCFILE7 has LOB data

Data Records in ASCFILE1:

```
1...5....10...15...20...25...30.
REC1 ASCFILE2 ASCFILE3
REC2 ASCFILE4 ASCFILE5
REC3 ASCFILE6 ASCFILE7
```

The following command loads the table from the file:

```
db2 load from ascfile1 of asc
   lobs from /u/user1, /u/user1/bin
   modified by lobsinfile reclen=22
   method L (1 4, 6 13, 15 22)
   insert into table1
```

**Notes:**

1. The specification of `lobsinfile` in the MODIFIED BY parameter tells the loader that all LOB data is to be loaded from files.
2. The specification of `reclen=22` in the MODIFIED BY parameter indicates that there is no new-line character at the end of each input record, and that each record is 22 bytes long.
3. LOB data is contained in 6 files, ASCFILE2 through ASCFILE7. Each file contains the data that will be used to load a LOB column for a specific row. The relationship between LOBs and other data is specified in ASCFILE1. The first record of this file tells the loader to place REC1 in COL1 of row 1. The contents of ASCFILE2 will be used to load LOB1 of

row 1, and the contents of ASCFILE3 will be used to load LOB2 of row 1. Similarly, ASCFILE4 and ASCFILE5 will be used to load LOB1 and LOB2 of row 2, and ASCFILE6 and ASCFILE7 will be used to load the LOBs of row 3.

4. The LOBS FROM parameter contains 2 paths that will be searched for the named LOB files when those files are required by the loader.
5. To load LOBs directly from ASCFILE1 (a non-delimited ASCII file), without the `lobsinfile` modifier, the following rules must be observed:
   - The total length of any record, including LOBs, cannot exceed 32KB.
   - LOB fields in the input records must be of fixed length, and LOB data padded with blanks as necessary.
   - The `striptblanks` modifier must be specified, so that the trailing blanks used to pad LOBs can be removed as the LOBs are inserted into the database.

**Example 3 (Using Dump Files)**

Table FRIENDS is defined as:

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

If an attempt is made to load the following data records into this table,

```
23, 24, bobby
, 45, john
4,, mary
```

the second row is rejected because the first INT is NULL, and the column definition specifies NOT NULL. Columns which contain initial characters that are not consistent with the DEL format will generate an error, and the record will be rejected. Such records can be written to a dump file (see Table 7 on page 359).

DEL data appearing in a column outside of character delimiters is ignored, but does generate a warning. For example:

```
22,34,"bob"
24,55,"sam" sdf
```

The utility will load ″sam″ in the third column of the table, and the characters ″sdf″ will be flagged in a warning. The record is not rejected. Another example:

```
22 3, 34,"bob"
```

The utility will load `22,34,"bob"`, and generate a warning that some data in column one following the `22` was ignored. The record is not rejected.

**Example 4 (Loading DATALINK Data)**

The following command loads the table MOVIETABLE from the input file delfile1, which has data in the DEL format:

```
db2 load from delfile1 of del
    modified by dldel|
    insert into movietable (actorname, description, url_making_of, url_movie)
    datalink specification (dl_url_default_prefix "http://narang"),
    (dl_url_replace_prefix "http://bomdel" dl_url_suffix ".mpeg")
    for exception excptab
```

**Notes:**

1. The table has four columns:
```
actorname              VARCHAR(n)
description            VARCHAR(m)
url_making_of          DATALINK (with LINKTYPE URL)
url_movie              DATALINK (with LINKTYPE URL)
```
2. The DATALINK data in the input file has the vertical bar (|) character as the sub-field delimiter.
3. If any column value for url_making_of does not have the prefix character sequence, ″http://narang″ is used.
4. Each non-NULL column value for url_movie will get ″http://bomdel″ as its prefix. Existing values are replaced.
5. Each non-NULL column value for url_movie will get ″.mpeg″ appended to the path. For example, if a column value of url_movie is ″http://server1/x/y/z″, it will be stored as ″http://bomdel/x/y/z.mpeg″; if the value is ″/x/y/z″, it will be stored as ″http://bomdel/x/y/z.mpeg″.
6. If any unique index or DATALINK exception occurs while loading the table, the affected records are deleted from the table and put into the exception table excptab.

## Usage Notes

Data is loaded in the sequence that appears in the input file. If a particular sequence is desired, the data should be sorted before a load is attempted.

The load utility builds indexes based on existing definitions. The exception tables are used to handle duplicates on unique keys. The utility does not enforce referential integrity, perform constraints checking, or update summary tables that are dependent on the tables being loaded. Tables that include referential or check constraints are placed in check pending state. Summary tables that are defined with REFRESH IMMEDIATE, and that are dependent on tables being loaded, are also placed in check pending state. Issue the SET INTEGRITY statement to take the tables out of check pending state. Load operations cannot be carried out on replicated summary tables.

If clustering is required, the data should be sorted on the clustering index prior to loading.

**DB2 Data Links Manager Considerations**

For each DATALINK column, there can be one column specification within parentheses. Each column specification consists of one or more of DL_LINKTYPE, *prefix* and a DL_URL_SUFFIX specification. The *prefix* information can be either DL_URL_REPLACE_PREFIX, or the DL_URL_DEFAULT_PREFIX specification.

There can be as many DATALINK column specifications as the number of DATALINK columns defined in the table. The order of specifications follows the order of DATALINK columns as found within the insert-column list (if specified by INSERT INTO (insert-column, ...)), or within the table definition (if insert-column is not specified).

For example, if a table has columns C1, C2, C3, C4, and C5, and among them only columns C2 and C5 are of type DATALINK, and the insert-column list is (C1, C5, C3, C2), there should be two DATALINK column specifications. The first column specification will be for C5, and the second column specification will be for C2. If an insert-column list is not specified, the first column specification will be for C2, and the second column specification will be for C5.

If there are multiple DATALINK columns, and some columns do not need any particular specification, the column specification should have at least the parentheses to unambiguously identify the order of specifications. If there are no specifications for any of the columns, the entire list of empty parentheses can be dropped. Thus, in cases where the defaults are satisfactory, there need not be any DATALINK specification.

If data is being loaded into a table with a DATALINK column that is defined with FILE LINK CONTROL, perform the following steps before invoking the load utility. (If all the DATALINK columns are defined with NO LINK CONTROL, these steps are not necessary).

1. Ensure that the DB2 Data Links Manager is installed on the Data Links servers that will be referred to by the DATALINK column values.
2. Ensure that the database is registered with the DB2 Data Links Manager.
3. Copy to the appropriate Data Links servers, all files that will be inserted as DATALINK values.
4. Define the prefix name (or names) to the DB2 Data Links Managers on the Data Links servers.
5. Register the Data Links servers referred to by DATALINK data (to be loaded) in the DB2 Data Links Manager configuration file.

The connection between DB2 and the Data Links server may fail while running the load utility, causing the load operation to fail. If this occurs:

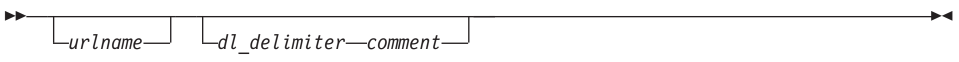
1. Start the Data Links server and the DB2 Data Links Manager.
2. Invoke a load restart operation.

Links that fail during the load operation are considered to be data integrity violations, and are handled in much the same way as unique index violations. Consequently, a special exception has been defined for loading tables that have one or more DATALINK columns. For additional information, refer to the description of exceptions in the *SQL Reference*.

**Representation of DATALINK Information in an Input File**

The LINKTYPE (currently only URL is supported) is not specified as part of DATALINK information. The LINKTYPE is specified in the LOAD or the IMPORT command, and for input files of type PC/IXF, in the appropriate column descriptor records as described in the "Export/Import/Load Utility File Formats" appendix in the *Data Movement Utilities Guide and Reference*.

The syntax of DATALINK information for a URL LINKTYPE is as follows:



Note that both *urlname* and *comment* are optional. If neither is provided, the NULL value is assigned.

**urlname**

The URL name must conform to valid URL syntax.

**Notes:**

1. Only "http" and "file" are permitted as a scheme name.
2. The prefix (scheme, host, and port) of the URL name is optional. If a prefix is not present, it is taken from the DL_URL_DEFAULT_PREFIX or the DL_URL_REPLACE_PREFIX specification of the load or the import utility. If none of these is specified, the prefix defaults to "file://localhost". Thus, in the case of local files, the file name with full path name can be entered as the URL name, without the need for a DATALINK column specification within the LOAD or the IMPORT command.
3. Prefixes, even if present in URL names, are overridden by a different prefix name on the DL_URL_REPLACE_PREFIX specification during a load or import operation.

4. The ″path″ (after appending DL_URL_SUFFIX, if specified) is the full path name of the remote file in the remote server. Relative path names are not allowed. The http server default path-prefix is not taken into account.

**dl_delimiter**
For the delimited ASCII (DEL) file format, a character specified via the `dldel` modifier, or defaulted to on the LOAD or the IMPORT command. For the non-delimited ASCII (ASC) file format, this should correspond to the character sequence `\;` (a backslash followed by a semicolon). Whitespace characters (blanks, tabs, and so on) are permitted before and after the value specified for this parameter.

**comment**
The comment portion of a DATALINK value. If specified for the delimited ASCII (DEL) file format, the *comment* text must be enclosed by the character string delimiter, which is double quotation marks (″) by default. This character string delimiter can be overridden by the MODIFIED BY *filetype-mod* specification of the LOAD or the IMPORT command.

If no comment is specified, the comment defaults to a string of length zero.

Following are DATALINK data examples for the delimited ASCII (DEL) file format:

- `http://www.almaden.ibm.com:80/mrep/intro.mpeg; "Intro Movie"`

  This is stored with the following parts:
  – scheme = http
  – server = www.almaden.ibm.com
  – path = /mrep/intro.mpeg
  – comment = ″Intro Movie″
- `file://narang/u/narang; "InderPal's Home Page"`

  This is stored with the following parts:
  – scheme = file
  – server = narang
  – path = /u/narang
  – comment = ″InderPal's Home Page″
- `file:/home/ff.gg; "hi there"`

  This is stored with the following parts:
  – scheme = file
  – server = localhost
  – path = /home/ff.gg

- comment = ″hi there″

Following are DATALINK data examples for the non-delimited ASCII (ASC) file format:

- `http://www.almaden.ibm.com:80/mrep/intro.mpeg\;Intro Movie`

  This is stored with the following parts:
  - scheme = http
  - server = www.almaden.ibm.com
  - path = /mrep/intro.mpeg
  - comment = ″Intro Movie″
- `file://narang/u/narang\; InderPal's Home Page`

  This is stored with the following parts:
  - scheme = file
  - server = narang
  - path = /u/narang
  - comment = ″InderPal's Home Page″
- `file:/home/ff.gg\; hi there`

  This is stored with the following parts:
  - scheme = file
  - server = localhost
  - path = /home/ff.gg
  - comment = ″hi there″

Following are DATALINK data examples in which the load or import specification for the column is assumed to be DL_URL_DEFAULT_PREFIX (″http://qso″):

- `file://narang/pics/xxx.jpeg?search_pat`

  This is stored with the following parts:
  - scheme = file
  - server = narang
  - path = /pics/xxx.jpeg
  - comment = NULL string
- `/u/me/myfile.ps`

  This is stored with the following parts:
  - scheme = http
  - server = qso
  - path = /u/me/myfile.ps
  - comment = NULL string

Table 7. Valid File Type Modifiers (LOAD)

| **Modifier** | **Description** |
|---|---|
| | **All File Formats** |
| anyorder | This modifier is used in conjunction with the *cpu_parallelism* parameter. Specifies that the preservation of source data order is not required, yielding significant additional performance benefit on SMP systems. If the value of *cpu_parallelism* is 1, this option is ignored. This option is not supported if SAVECOUNT > 0, since crash recovery after a consistency point requires that data be loaded in sequence. |
| fastparse | Reduced syntax checking is done on user-supplied column values, and performance is enhanced. Tables loaded under this option are guaranteed to be architecturally correct, and the utility is guaranteed to perform sufficient data checking to prevent a segmentation violation or trap. Data that is in correct form will be loaded correctly.<br><br>For example, if a value of `123qwr4` were to be encountered as a field entry for an integer column in an ASC file, the load utility would ordinarily flag a syntax error, since the value does not represent a valid number. With `fastparse`, a syntax error is not detected, and an arbitrary number is loaded into the integer field. Care must be taken to use this modifier with clean data only. Performance improvements using this option with ASCII data can be quite substantial, but `fastparse` does not significantly enhance performance with PC/IXF data, since IXF is a binary format, and `fastparse` affects parsing and conversion from ASCII to internal forms. |
| indexfreespace=*x* | *x* is an integer between 0 and 99 inclusive. The value is interpreted as the percentage of each index page that is to be left as free space when loading the index. The first entry in a page is added without restriction; subsequent entries are added if the percent free space threshold can be maintained. The default value is the one used at CREATE INDEX time.<br><br>This value takes precedence over the PCTFREE value specified in the CREATE INDEX statement, and affects index leaf pages only. |
| lobsinfile | *lob-path* specifies the path to the files containing LOB values. The ASC, DEL, or IXF load input files contain the names of the files having LOB data in the LOB column. |

Table 7. Valid File Type Modifiers (LOAD) (continued)

| **Modifier** | **Description** |
|---|---|
| noheader | Skips the header verification code.<br><br>The AutoLoader utility (see “AutoLoader” in the *Data Movement Utilities Guide and Reference*) writes a header to each file contributing data to a table in a multi-node nodegroup. The header includes the node number, the partitioning map, and the partitioning key specification. The load utility requires this information to verify that the data is being loaded at the correct node. When loading files into a table that exists on a single-node nodegroup, the headers do not exist, and this option causes the load utility to skip the header verification code. |
| norowwarnings | Suppresses all warnings about rejected rows. |
| pagefreespace=*x* | *x* is an integer between 0 and 100 inclusive. The value is interpreted as the percentage of each data page that is to be left as free space.<br><br>If the specified value is invalid because of the minimum row size, (for example, a row that is at least 3 000 bytes long, and an *x* value of 50), the row will be placed on a new page. If a value of 100 is specified, each row will reside on a new page.<br>**Note:** The PCTFREE value of a table determines the amount of free space designated per page. If a `pagefreespace` value on the load operation or a PCTFREE value on a table have not been set, the utility will fill up as much space as possible on each page. The value set by `pagefreespace` overrides the PCTFREE value specified for the table. |
| totalfreespace=*x* | *x* is an integer between 0 and 100 inclusive. The value is interpreted as the percentage of the total pages in the table that is to be appended to the end of the table as free space. For example, if *x* is 20, and the table has 100 data pages, 20 additional empty pages will be appended. The total number of data pages for the table will be 120. |

Table 7. Valid File Type Modifiers (LOAD) (continued)

| Modifier | Description |
|---|---|
| usedefaults | If a source column for a target table column has been specified, but it contains no data for one or more row instances, default values are loaded. Examples of missing data are:<br>• For DEL files: ",," is specified for the column<br>• For DEL/ASC/WSF files: A row that does not have enough columns, or is not long enough for the original specification.<br><br>Without this option, if a source column contains no data for a row instance, one of the following occurs:<br>• If the column is nullable, a NULL is loaded<br>• If the column is not nullable, the utility rejects the row. |
| **ASCII File Formats (ASC/DEL)** | |
| codepage=*x* | *x* is an ASCII character string. The value is interpreted as the code page of the data in the input data set. Converts character data (and numeric data specified in characters) from this code page to the database code page during the load operation.<br><br>The following rules apply:<br>• For pure DBCS (graphic), mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive.<br>• For DEL data specified in an EBCDIC code page, the delimiters may not coincide with the shift-in and shift-out DBCS characters.<br>• `nullindchar` must specify symbols included in the standard ASCII set between code points x20 and x7F, inclusive. This refers to ASCII symbols and code points. EBCDIC data can use the corresponding symbols, even though the code points will be different. |

Table 7. Valid File Type Modifiers (LOAD) (continued)

| **Modifier** | **Description** |
|---|---|
| dumpfile = *x* | *x* is the fully qualified (according to the server node) name of an exception file to which rejected rows are written. A maximum of 32KB of data is written per record. Following is an example that shows how to specify a dump file:<br>`db2 load from data of del`<br>`   modified by dumpfile = /u/user/filename`<br>`   insert into table_name`<br>**Notes:**<br>1. In a partitioned database environment, the path should be local to the loading node, so that concurrently running load operations do not attempt to write to the same file.<br>2. The contents of the file are written to disk in an asynchronous buffered mode. In the event of a failed or an interrupted load operation, the number of records committed to disk cannot be known with certainty, and consistency cannot be guaranteed after a LOAD RESTART. The file can only be assumed to be complete for a load operation that starts and completes in a single pass.<br>3. This modifier does not support file names with multiple file extensions. For example,<br>`dumpfile = /home/svtdbm6/DUMP.FILE`<br>is acceptable to the load utility, but<br>`dumpfile = /home/svtdbm6/DUMP.LOAD.FILE`<br>is not. |
| implieddecimal | The location of an implied decimal point is determined by the column definition; it is no longer assumed to be at the end of the value. For example, the value `12345` is loaded into a DECIMAL(8,2) column as `123.45`, *not* `12345.00`. |
| noeofchar | The optional end-of-file character `x'1A'` is not recognized as the end of file. Processing continues as if it were a normal character. |
| **ASC (Non-delimited ASCII) File Format** | |

Table 7. Valid File Type Modifiers (LOAD) (continued)

| **Modifier** | **Description** |
|---|---|
| binarynumerics | Numeric (but not DECIMAL) data must be in binary form, not the character representation. This avoids costly conversions.<br><br>This option is supported only with positional ASC, using fixed length records specified by the `reclen` option. The `noeofchar` option is assumed.<br><br>The following rules apply:<br>• No conversion between data types is performed, with the exception of BIGINT, INTEGER, and SMALLINT.<br>• Data lengths must match their target column definitions.<br>• FLOATs must be in IEEE Floating Point format.<br>• Binary data in the load source file is assumed to be big-endian, regardless of the platform on which the load operation is running.<br><br>**Note:** NULLs cannot be present in the data for columns affected by this modifier. Blanks (normally interpreted as NULL) are interpreted as a binary value when this modifier is used. |
| nochecklengths | If `nochecklengths` is specified, an attempt is made to load each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully loaded if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions. |
| nullindchar=*x* | *x* is a single character. Changes the character denoting a NULL value to *x*. The default value of *x* is `Y`.[b]<br><br>This modifier is case sensitive for EBCDIC data files, except when the character is an English letter. For example, if the NULL indicator character is specified to be the letter `N`, then `n` is also recognized as a NULL indicator. |

Table 7. Valid File Type Modifiers (LOAD) (continued)

| **Modifier** | **Description** |
|---|---|
| packeddecimal | Loads packed-decimal data directly, since the `binarynumerics` modifier does not include the DECIMAL field type.<br><br>This option is supported only with positional ASC, using fixed length records specified by the `reclen` option. The `noeofchar` option is assumed.<br><br>Supported values for the sign nibble are:<br>`+ = 0xC 0xA 0xE 0xF`<br>`- = 0xD 0xB`<br><br>**Note:** NULLs cannot be present in the data for columns affected by this modifier. Blanks (normally interpreted as NULL) are interpreted as a binary value when this modifier is used.<br><br>Regardless of the server platform, the byte order of binary data in the load source file is assumed to be big-endian; that is, when using this modifier on OS/2 or on the Windows operating system, the byte order must not be reversed. |
| reclen=*x* | *x* is an integer with a maximum value of 32 767. *x* characters are read for each row, and a new-line character is not used to indicate the end of the row. |
| striptblanks | Truncates any trailing blank spaces when loading data into a variable-length field. If this option is not specified, blank spaces are kept.<br><br>This option cannot be specified together with `striptnulls`. These are mutually exclusive options.<br>**Note:** This option replaces the obsolete `t` option, which is supported for back-level compatibility only. |
| striptnulls | Truncates any trailing NULLs (0x00 characters) when loading data into a variable-length field. If this option is not specified, NULLs are kept.<br><br>This option cannot be specified together with `striptblanks`. These are mutually exclusive options.<br>**Note:** This option replaces the obsolete `padwithzero` option, which is supported for back-level compatibility only. |
| **DEL (Delimited ASCII) File Format** | |

Table 7. Valid File Type Modifiers (LOAD) (continued)

| **Modifier** | **Description** |
|---|---|
| chardel*x* | *x* is a single character string delimiter. The default value is a double quotation mark ("). The specified character is used in place of double quotation marks to enclose a character string.[ab]<br><br>The single quotation mark (') can also be specified as a character string delimiter as follows:<br>`modified by chardel''` |
| coldel*x* | *x* is a single character column delimiter. The default value is a comma (,). The specified character is used in place of a comma to signal the end of a column.[ab] |
| datesiso | Date format. Causes all date data values to be loaded in ISO format. |
| decplusblank | Plus sign character. Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign. |
| decpt*x* | *x* is a single character substitute for the period as a decimal point character. The default value is a period (.). The specified character is used in place of a period as a decimal point character.[ab] |
| delprioritychar | The current default priority for delimiters is: record delimiter, character delimiter, column delimiter. This modifier protects existing applications that depend on the older priority by reverting the delimiter priorities to: character delimiter, record delimiter, column delimiter. Syntax:<br>`db2 load ... modified by delprioritychar ...`<br><br>For example, given the following DEL data file:<br>`"Smith, Joshua",4000,34.98<row delimiter>`<br>`"Vincent,<row delimiter>, is a manager", ...`<br>`... 4005,44.37<row delimiter>`<br><br>With the `delprioritychar` modifier specified, there will be only two rows in this data file. The second <row delimiter> will be interpreted as part of the first data column of the second row, while the first and the third <row delimiter> are interpreted as actual record delimiters. If this modifier is *not* specified, there will be three rows in this data file, each delimited by a <row delimiter>. |

Table 7. Valid File Type Modifiers (LOAD) (continued)

| **Modifier** | **Description** |
|---|---|
| dldel*x* | *x* is a single character DATALINK delimiter. The default value is a semicolon (;). The specified character is used in place of a semicolon as the inter-field separator for a DATALINK value. It is needed because a DATALINK value may have more than one sub-value. [abc]<br>**Note:** *x* must not be the same character specified as the row, column, or character string delimiter. |
| nodoubledel | Suppresses recognition of double character delimiters. |
| | **IXF File Format** |
| forcein | Directs the utility to accept data despite code page mismatches, and to suppress translation between code pages.<br><br>Fixed length target fields are checked to verify that they are large enough for the data. If `nochecklengths` is specified, no checking is done, and an attempt is made to load each row. |
| nochecklengths | If `nochecklengths` is specified, an attempt is made to load each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully loaded if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions. |

**Notes:**

1. The load utility does not issue a warning if an attempt is made to use unsupported file types with the MODIFIED BY option. If this is attempted, the load operation fails, and an error code is returned.
2. [a] “Delimiter Restrictions” on page 213 lists restrictions that apply to the characters that can be used as delimiter overrides.
3. [b] The character must be specified in the code page of the source data.

   The character code point (instead of the character symbol), can be specified using the syntax xJJ or 0xJJ, where JJ is the hexadecimal representation of the code point. For example, to specify the # character as a column delimiter, use one of the following:

   ```
   ... modified by coldel# ...
   ... modified by coldel0x23 ...
   ... modified by coldelX23 ...
   ```
4. [c] Even if the DATALINK delimiter character is a valid character within the URL syntax, it will lose its special meaning within the scope of the load operation.

## See Also

"LOAD QUERY" on page 368

"QUIESCE TABLESPACES FOR TABLE" on page 399.

## LOAD QUERY

Checks the status of a load operation during processing. A connection to the same database, and a separate CLP session are also required to successfully invoke this command. It can be used either by local or remote users.

### Authorization

None

### Required Connection

Database

### Command Syntax

```
>>──LOAD QUERY──TABLE──table-name──┬──────────────────────┬──┬─────────────┬──────────>
                                   └─TO──local-message-file─┘  ├─NOSUMMARY───┤
                                                               └─SUMMARYONLY─┘

>──┬───────────┬───────────────────────────────────────────────────────────────────><
   └─SHOWDELTA─┘
```

### Command Parameters

**NOSUMMARY**
Specifies that no load summary information (rows read, rows skipped, rows loaded, rows rejected, rows deleted, rows committed, and number of warnings) is to be reported.

**SHOWDELTA**
Specifies that only new information (pertaining to load events that have occurred since the last invocation of the LOAD QUERY command) is to be reported.

**SUMMARYONLY**
Specifies that only load summary information is to be reported.

**TABLE table-name**
Specifies the name of the table into which data is currently being loaded.

**TO local-message-file**
Specifies the destination for warning and error messages that occur during the load operation. This file cannot be the *message-file* specified for the LOAD command. If the file already exists, all messages that the load utility has generated are appended to it.

## Examples

A user loading a large amount of data into the STAFF table wants to check the status of the load operation. The user can specify:

```
   db2 connect to <database>
   db2 load query table staff to /u/mydir/staff.tempmsg
```

The output file `/u/mydir/staff.tempmsg` might look like the following:

```
SQL3500W The utility is beginning the "LOAD" phase at time
"02-13-1997 19:40:29.645353".

SQL3519W  Begin Load Consistency Point. Input record count = "0".

SQL3520W  Load Consistency Point was successful.

SQL3109N The utility is beginning to load data from file
"/u/mydir/data/staffbig.ixf".

SQL3150N The H record in the PC/IXF file has product "DB2 01.00",
date "19970111", and time "194554".

SQL3153N The T record in the PC/IXF file has name
"data/staffbig.ixf", qualifier " ", and source " ".

SQL3519W  Begin Load Consistency Point. Input record count =
"111152".

SQL3520W  Load Consistency Point was successful.

SQL3519W  Begin Load Consistency Point. Input record count =
"222304".

SQL3520W  Load Consistency Point was successful.
```

## See Also

"LOAD" on page 338.

## MIGRATE DATABASE

Converts previous (Version 2.x or higher) versions of DB2 databases to current formats.

**Attention:** The database pre-migration tool, “db2ckmig - Database Pre-migration Tool” on page 23, must be run prior to DB2 Version 6 installation (on OS/2 or the Windows operating system), or before instance migration (on UNIX based systems), because it cannot be executed on DB2 Version 6. Backup all databases prior to migration, and prior to DB2 Version 6 installation on OS/2 or the Windows operating system.

For detailed information about database migration, see one of the *Quick Beginnings* books.

### Authorization

*sysadm*

### Required Connection

This command establishes a database connection.

### Command Syntax

```
►►──MIGRATE──┬─DATABASE─┬──database-alias──────────────────────────────────►
             └─DB───────┘

►──┬─────────────────────────────────┬──────────────────────────────────►◄
   └─USER──username──┬─────────────────┬─┘
                     └─USING──password─┘
```

### Command Parameters

**DATABASE database-alias**
: Specifies the alias of the database to be migrated to the currently installed version of the database manager.

**USER username**
: Identifies the user name under which the database is to be migrated.

**USING password**
: The password used to authenticate the user name. If the password is omitted, but a user name was specified, the user is prompted to enter it.

## Examples

The following example migrates the database cataloged under the database alias `sales`:

```
db2 migrate database sales
```

## Usage Notes

This command will only migrate a database to a newer version, and cannot be used to convert a migrated database to its previous version.

The database must be cataloged before migration.

## PRECOMPILE PROGRAM

Processes an application program source file containing embedded SQL statements. A modified source file is produced, containing host language calls for the SQL statements and, by default, a package is created in the database.

### Scope

This command can be issued from any node in `db2nodes.cfg`. It updates the database catalogs on the catalog node. Its effects are visible to all nodes.

### Authorization

One of the following:

- *sysadm* or *dbadm* authority
- BINDADD privilege if a package does not exist, and one of:
  - IMPLICIT_SCHEMA authority on the database if the schema name of the package does not exist
  - CREATEIN privilege on the schema if the schema name of the package exists
- ALTERIN privilege on the schema if the package exists
- BIND privilege on the package if it exists.

The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements. If the user has *sysadm* authority, but not explicit privileges to complete the bind, the database manager grants explicit *dbadm* authority automatically.

### Required Connection

Database. If implicit connect is enabled, a connection to the default database is established.

### Command Syntax

**For DB2**

```
►►──┬─PRECOMPILE─┬──filename──┬──────────────────────────┬──────────────────►
    └─PREP───────┘            └─BINDFILE─┬──────────────┬─┘
                                         └─USING─bind-file─┘
```

```
►──┬──────────────────────────────┬──┬──────────────────────────────┬──┬─────────────────────┬──────────►
   │               ┌─UNAMBIG─┐    │  └─COLLECTION──schema-name──────┘  │              ┌─1─┐  │
   └─BLOCKING──────┼─ALL─────┼────┘                                    └─CONNECT──────┴─2─┴──┘
                   └─NO──────┘

►──┬─────────────────────────┬──┬─────────────────────────────────────┬─────────────────────────────────►
   │             ┌─DEF─┐     │  │                       ┌─NO──┐       │
   └─DATETIME────┼─EUR─┼─────┘  └─DEFERRED_PREPARE──────┼─ALL─┼───────┘
                 ├─ISO─┤                                └─YES─┘
                 ├─JIS─┤
                 ├─LOC─┤
                 └─USA─┘

►──┬───────────────────────────────────────────┬──┬────────────────────────────────────┬────────────────►
   │          ┌─1──────────────────────┐       │  │               ┌─EXPLICIT────┐      │
   └─DEGREE───┼─degree-of-parallelism──┼───────┘  └─DISCONNECT────┼─AUTOMATIC───┼──────┘
              └─ANY────────────────────┘                          └─CONDITIONAL─┘

►──┬─────────────────────────────┬──┬─────────────────────────┬──┬─────────────────────────┬────────────►
   │                  ┌─RUN──┐   │  │            ┌─NO──┐      │  │             ┌─NO──┐     │
   └─DYNAMICRULES─────┴─BIND─┴───┘  └─EXPLAIN────┼─ALL─┼──────┘  └─EXPLSNAP────┼─ALL─┼─────┘
                                                 └─YES─┘                       └─YES─┘

►──┬────────────────────────────────────────┬──┬───────────────────────┬──┬───────────────────────┬─────►
   │              ┌─,────────────────┐      │  │          ┌─DEF─┐      │  │               ┌─CS─┐  │
   │              ▼                  │      │  └─INSERT───┴─BUF─┴──────┘  └─ISOLATION─────┼─RR─┼──┘
   └─FUNCPATH─────┴──schema-name─────┴──────┘                                             ├─RS─┤
                                                                                          └─UR─┘

►──┬───────────────────────────┬──┬──────────────────────────────┬─────────────────────────────────────►
   │                ┌─SAA1───┐ │  └─MESSAGES──message-file───────┘
   └─LANGLEVEL──────┼─MIA────┼─┘
                    └─SQL92E─┘

►──┬───────────────┬──┬────────────────────────┬──┬─────────────────────────┬──────────────────────────►
   └─NOLINEMACRO───┘  │                 ┌─0─┐  │  └─OUTPUT──filename────────┘
                      └─OPTLEVEL────────┴─1─┴──┘

►──┬────────────────────────────────┬──┬─────────────────────────────────────┬────────────────────────►
   └─OWNER──authorization-id────────┘  └─PACKAGE──┬───────────────────────┬──┘
                                                  └─USING──package-name───┘
```

```
►──┬─────────────────────────────────────────────────┬──┬──────────────────────────────┬──────────►
   └─PREPROCESSOR──┬─"preprocessor-command"─┬──────┘  └─QUALIFIER──qualifier-name──┘
                   └─'preprocessor-command'─┘

►──┬──────────────────────────────────┬──┬──────────────────────┬─────────────────────────────────►
   └─QUERYOPT──optimization-level─────┘  │         ┌─SAA──┐     │
                                         └─SQLCA───┴─NONE─┴─────┘

►──┬──────────────────────────────────────┬──┬─────────────────────────────────────────┬──────────►
   │   (1)              ┌─NOPACKAGE─┐     │  └─SQLFLAG──┬─SQL92E─────┬──SYNTAX────────┘
   └──────────SQLERROR──┴─CHECK─────┴─────┘             ├─MVSDB2V23──┤
                                                        ├─MVSDB2V31──┤
                                                        └─MVSDB2V41──┘

►──┬─────────────────────────┬──┬────────────────────────┬──┬─────────────────────────────────┬───►
   │         ┌─DB2──┐        │  │         ┌─YES──┐       │  │            ┌─ONEPHASE──┐        │
   └─SQLRULES┴─STD──┴────────┘  └─SQLWARN─┴─NO───┴───────┘  └─SYNCPOINT──┼─NONE──────┤────────┘
                                                                         └─TWOPHASE──┘

►──┬──────────┬──┬──────────────────────────────────┬──┬─────────────────────────────────┬───────►◄
   └─SYNTAX───┘  │          ┌─IBMCOB──────────────┐ │  │            ┌─NOCONVERT─┐        │
                 └─TARGET───┼─MFCOB───────────────┼─┘  └─WCHARTYPE──┴─CONVERT───┴────────┘
                            ├─MFCOB16─────────────┤
                            ├─ANSI_COBOL──────────┤
                            ├─C───────────────────┤
                            ├─CPLUSPLUS───────────┤
                            ├─FORTRAN─────────────┤
                            ├─BORLAND_C───────────┤
                            └─BORLAND_CPLUSPLUS───┘
```

**Notes:**

1. SYNTAX is a synonym for SQLERROR(CHECK).

**For DRDA**

```
►►──┬─PRECOMPILE─┬──filename──────────────────────────────────────────────────────────────────────►
    └─PREP───────┘

►──┬────────────────────────────────────────────────────────────────────────────────────┬─────────►
   └─ACTION──┬─ADD──────────────────────────────────────────────────────────────────┬───┘
             │ ┌─REPLACE─┐                                                          │
             └─┴─────────┴──┬─────────────────────────┬──┬──────────────────────┬───┘
                            │          ┌─YES─┐        │  └─REPLVER──version-id──┘
                            └─RETAIN───┴─NO──┴────────┘
```

```
►──┬──────────────────────────────────┬──┬────────────────────────────┬────────────►
   └─BINDFILE──┬────────────────────┬─┘  └─BLOCKING──┬─UNAMBIG─┬─────┘
               └─USING──bind-file───┘                ├─ALL─────┤
                                                     └─NO──────┘

►──┬────────────────────────┬──┬───────────────────────┬──┬──────────────────────┬──►
   └─CCSIDG──double-ccsid───┘  └─CCSIDM──mixed-ccsid───┘  └─CCSIDS──sbcs-ccsid───┘

►──┬──────────────────────────────┬──┬────────────────────────┬──┬──────────────────────────────┬──►
   └─CHARSUB──┬─DEFAULT─┬─────────┘  └─CNULREQD──┬─YES─┬──────┘  └─COLLECTION──schema-name─────┘
              ├─BIT─────┤                        └─NO──┘
              ├─MIXED───┤
              └─SBCS────┘

►──┬─────────────────────┬──┬─────────────────────────────┬──┬──────────────────┬──►
   └─CONNECT──┬─1─┬──────┘  │  (1)                        │  └─DEC──┬─15─┬─────┘
              └─2─┘         └─────────DATETIME──┬─DEF─┬───┘         └─31─┘
                                                ├─EUR─┤
                                                ├─ISO─┤
                                                ├─JIS─┤
                                                ├─LOC─┤
                                                └─USA─┘

►──┬────────────────────────────┬──┬─────────────────────────────────────┬──────────►
   └─DECDEL──┬─PERIOD─┬─────────┘  └─DEFERRED_PREPARE──┬─NO──┬───────────┘
             └─COMMA──┘                                ├─ALL─┤
                                                       └─YES─┘

►──┬──────────────────────────────────────────────────────┬──┬──────────────────────────────────┬──►
   │  (2)                                                 │  └─DISCONNECT──┬─EXPLICIT────┬──────┘
   └─────────DEGREE──┬─1───────────────────────┬──────────┘                ├─AUTOMATIC───┤
                     ├─degree-of-parallelism───┤                           └─CONDITIONAL─┘
                     └─ANY─────────────────────┘

►──┬─────────────────────────────────┬──┬─────────────────────────┬──┬─────────────────────┬──►
   └─DYNAMICRULES──┬─RUN────┬────────┘  └─EXPLAIN──┬─NO──┬────────┘  └─GENERIC──string─────┘
                   ├─BIND───┤                      └─YES─┘
                   ├─DEFINE─┤
                   └─INVOKE─┘

►──┬──────────────────────────┬──┬───────────────────────────────┬────────────────►
   └─ISOLATION──┬─CS─┬────────┘  └─LEVEL──consistency-token──────┘
                ├─NC─┤
                ├─RR─┤
                ├─RS─┤
                └─UR─┘
```

```
►─┬──────────────────────────┬─┬─────────────┬─┬───────────────────┬──────────────►
  └─MESSAGES──message-file───┘ └─NOLINEMACRO─┘ │           ┌─0─┐   │
                                               └─OPTLEVEL──┴─1─┴───┘

►─┬───────────────────────────┬─┬────────────────────────────────┬────────────────►
  └─OWNER──authorization-id───┘ └─PACKAGE─┬────────────────────┬─┘
                                          └─USING──package-name─┘

►─┬─────────────────────────────────────────────────┬─┬────────────────────────────┬─►
  └─PREPROCESSOR──┬─"preprocessor-command"─┬───────┘ └─QUALIFIER──qualifier-name───┘
                  └─'preprocessor-command'─┘

►─┬──────────────────────────────┬─┬──────────────────────────────────────┬──────►
  │           ┌─COMMIT─────┐     │ │  (3)             ┌─NOPACKAGE─┐       │
  └─RELEASE───┴─DEALLOCATE─┴─────┘ └─────────SQLERROR─┼─CHECK─────┼───────┘
                                                      └─CONTINUE──┘

►─┬─────────────────────────────────────┬─┬─────────────────────────┬────────────►
  └─SQLFLAG──┬─SQL92E──────┬──SYNTAX────┘ │            ┌─DB2─┐      │
             ├─MVSDB2V23───┤              └─SQLRULES───┴─STD─┴──────┘
             ├─MVSDB2V31───┤
             └─MVSDB2V41───┘

►─┬──────────────────────────────┬─┬────────────────────────────────┬────────────►
  │          ┌─APOSTROPHE─┐      │ │             ┌─ONEPHASE─┐       │
  └─STRDEL───┴─QUOTE──────┴──────┘ └─SYNCPOINT───┼─NONE─────┼───────┘
                                                 └─TWOPHASE─┘

►──┬──────────┬─┬────────────────────────────────────┬─┬────────────────┬───────►
   └─SYNTAX───┘ └─TARGET───┬─IBMCOB────────────────┬─┘ └─TEXT──label────┘
                           ├─MFCOB─────────────────┤
                           ├─MFCOB16───────────────┤
                           ├─ANSI_COBOL────────────┤
                           ├─C─────────────────────┤
                           ├─CPLUSPLUS─────────────┤
                           ├─FORTRAN───────────────┤
                           ├─BORLAND_C─────────────┤
                           └─BORLAND_CPLUSPLUS─────┘

►─┬───────────────────────┬─┬──────────────────────┬─┬────────────────────────────────┬──►◄
  │           ┌─RUN──┐    │ └─VERSION──version-id──┘ │             ┌─NOCONVERT─┐      │
  └─VALIDATE──┴─BIND─┴────┘                          └─WCHARTYPE───┴─CONVERT───┴──────┘
```

**Notes:**

1. The DATETIME DEF option is not supported by DRDA, and is mapped to ISO when going through DB2 Connect.
2. The DEGREE option is only supported by DRDA Level 2 Application Servers.
3. SYNTAX is a synonym for SQLERROR(CHECK).

## Command Parameters

**filename**
Specifies the source file to be precompiled. An extension of:

- `.sqc` must be specified for C applications (generates a `.c` file)
- `.sqx` (OS/2 or the Windows operating system), or `.sqC` (UNIX based systems) must be specified for C++ applications (generates a `.cxx` file on OS/2 or the Windows operating system, or a `.C` file on UNIX based systems)
- `.sqb` must be specified for COBOL applications (generates a `.cbl` file)
- `.sqf` must be specified for FORTRAN applications (generates a `.for` file on OS/2 or the Windows operating system, or a `.f` file on UNIX based systems).

The preferred extension for C++ applications containing embedded SQL on UNIX based systems is `sqC`; however, the `sqx` convention, which was invented for systems that are not case sensitive, is tolerated by UNIX based systems.

**ACTION**
Indicates whether the package can be added or replaced. This DRDA precompile/bind option is not supported by DB2.

**ADD** Indicates that the named package does not exist, and that a new package is to be created. If the package already exists, execution stops, and a diagnostic error message is returned.

**REPLACE**
Indicates that the old package is to be replaced by a new one with the same location, collection, and package name.

**RETAIN**
Indicates whether EXECUTE authorities are to be preserved when a package is replaced. If ownership of the package changes, the new owner grants the BIND and EXECUTE authority to the previous package owner.

**NO** Does not preserve EXECUTE authorities when a package is replaced.

**YES** Preserves EXECUTE authorities when a package is replaced.

**REPLVER version-id**
Replaces a specific version of a package. The version identifier specifies which version of the package is to be replaced. Maximum length is 254 characters.

**BINDFILE**
Results in the creation of a bind file. A package is not created unless the **package** option is also specified. If a bind file is requested, but no package is to be created, as in the following example:

```
db2 prep sample.sqc bindfile
```

object existence and authentication SQLCODEs will be treated as warnings instead of errors. This will allow a bind file to be successfully created, even if the database being used for precompilation does not have all of the objects referred to in static SQL statements within the application. The bind file can be successfully bound, creating a package, once the required objects have been created.

**USING bind-file**
The name of the bind file that is to be generated by the precompiler. The file name must have an extension of `.bnd`. If a file name is not entered, the precompiler uses the name of the program (entered as the *filename* parameter), and adds the `.bnd` extension. If a path is not provided, the bind file is created in the current directory.

**BLOCKING**
For information about row blocking, see the *Administration Guide* or the *Application Development Guide.*

**ALL** Specifies to block for:
- Read-only cursors
- Cursors not specified as FOR UPDATE OF

Ambiguous cursors are treated as read-only.

**NO** Specifies not to block any cursors. Ambiguous cursors are treated as updatable.

**UNAMBIG**
Specifies to block for:
- Read-only cursors
- Cursors not specified as FOR UPDATE OF

Ambiguous cursors are treated as updatable.

**CCSIDG double-ccsid**
An integer specifying the coded character set identifier (CCSID) to be used for double byte characters in character column definitions (without a specific CCSID clause) in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**CCSIDM mixed-ccsid**
An integer specifying the coded character set identifier (CCSID) to be used for mixed byte characters in character column definitions (without a specific CCSID clause) in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**CCSIDS sbcs-ccsid**
An integer specifying the coded character set identifier (CCSID) to be used for single byte characters in character column definitions (without a specific CCSID clause) in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**CHARSUB**
Designates the default character sub-type that is to be used for column definitions in CREATE and ALTER TABLE SQL statements. This DRDA precompile/bind option is not supported by DB2.

**BIT** Use the FOR BIT DATA SQL character sub-type in all new character columns for which an explicit sub-type is not specified.

**DEFAULT**
Use the target system defined default in all new character columns for which an explicit sub-type is not specified.

**MIXED**
Use the FOR MIXED DATA SQL character sub-type in all new character columns for which an explicit sub-type is not specified.

**SBCS** Use the FOR SBCS DATA SQL character sub-type in all new character columns for which an explicit sub-type is not specified.

**CNULREQD**
This option is related to the **langlevel** precompile option, which is not

supported by DRDA. It is valid only if the bind file is created from a C or a C++ application. This DRDA bind option is not supported by DB2.

**NO** The application was coded on the basis of the **langlevel** SAA1 precompile option with respect to the null terminator in C string host variables.

**YES** The application was coded on the basis of the **langlevel** MIA precompile option with respect to the null terminator in C string host variables.

**COLLECTION schema-name**
Specifies an 8-character collection identifier for the package. If not specified, the authorization identifier for the user processing the package is used.

**CONNECT**

**1** Specifies that a CONNECT statement is to be processed as a type 1 CONNECT.

**2** Specifies that a CONNECT statement is to be processed as a type 2 CONNECT.

**DATETIME**
Specifies the date and time format to be used. For more information about date and time formats, see the *SQL Reference*.

**DEF** Use a date and time format associated with the country code of the database.

**EUR** Use the IBM standard for Europe date and time format.

**ISO** Use the date and time format of the International Standards Organization.

**JIS** Use the date and time format of the Japanese Industrial Standard.

**LOC** Use the date and time format in local form associated with the country code of the database.

**USA** Use the IBM standard for U.S. date and time format.

**DEC** Specifies the maximum precision to be used in decimal arithmetic operations. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**15** 15-digit precision is used in decimal arithmetic operations.

**31** 31-digit precision is used in decimal arithmetic operations.

**DECDEL**
Designates whether a period (.) or a comma (,) will be used as the decimal point indicator in decimal and floating point literals. This DRDA precompile/bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**COMMA**
Use a comma (,) as the decimal point indicator.

**PERIOD**
Use a period (.) as the decimal point indicator.

**DEFERRED_PREPARE**
Provides a performance enhancement when accessing DB2 common server databases or DRDA databases. This option combines the SQL PREPARE statement flow with the associated OPEN, DESCRIBE, or EXECUTE statement flow to minimize inter-process or network flow.

**NO** The PREPARE statement will be executed at the time it is issued.

**YES** Execution of the PREPARE statement will be deferred until the corresponding OPEN, DESCRIBE, or EXECUTE statement is issued.

The PREPARE statement will not be deferred if it uses the INTO clause, which requires an SQLDA to be returned immediately. However, if the PREPARE INTO statement is issued for a cursor that does not use any parameter markers, the processing will be optimized by pre-OPENing the cursor when the PREPARE is executed.

**ALL** Same as YES, except that a PREPARE INTO statement is also deferred. If the PREPARE statement uses the INTO clause to return an SQLDA, the application must not reference the content of this SQLDA until the OPEN, DESCRIBE, or EXECUTE statement is issued and returned.

**DEGREE**
Specifies the degree of parallelism for the execution of static SQL statements in an SMP system. This option does not affect CREATE INDEX parallelism.

**1** The execution of the statement will not use parallelism.

**degree-of-parallelism**
Specifies the degree of parallelism with which the statement can be executed, a value between 2 and 32 767 (inclusive).

**ANY** Specifies that the execution of the statement can involve parallelism using a degree determined by the database manager.

**DISCONNECT**

**AUTOMATIC**
Specifies that all database connections are to be disconnected at commit.

**CONDITIONAL**
Specifies that the database connections that have been marked RELEASE or have no open WITH HOLD cursors are to be disconnected at commit.

**EXPLICIT**
Specifies that only database connections that have been explicitly marked for release by the RELEASE statement are to be disconnected at commit.

**DYNAMICRULES**
Defines which rules apply to dynamic SQL at run time for the initial setting of the values used for authorization ID and for the implicit qualification of unqualified object references.

**RUN** Specifies that the authorization ID of the user executing the package is to be used. This is the default value.

**BIND** Specifies that all of the rules that apply to static SQL for authorization and qualification are to be used at run time. That is, the authorization ID of the package owner is to be used for authorization checking of dynamic SQL statements, and the default package qualifier is to be used for implicit qualification of unqualified object references within dynamic SQL statements.

When binding a package with this option, the binder of the package should not have any authorities that the user of the package should not receive, because dynamic SQL statements will be using the authorization ID of the package owner. The following dynamically prepared SQL statements cannot be used within a package that has been bound with this option: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET CONSTRAINTS, and SET EVENT MONITOR STATE.

**DEFINE**
Indicates that the authorization identifier used for the execution of dynamic SQL statements in a UDF or stored

procedure is the definer of the UDF or stored procedure. This option is not supported by DB2.

**INVOKE**
Indicates that the authorization identifier used for the execution of dynamic SQL statements in a UDF or stored procedure is the invoker of the UDF or stored procedure. This option is not supported by DB2.

**EXPLAIN**
Stores information in the Explain tables about the access plans chosen for each SQL statement in the package. DRDA does not support the ALL value for this option.

**NO** Explain information will not be captured.

**YES** Explain tables will be populated with information about the chosen access plan.

**ALL** Explain information for each eligible static SQL statement will be placed in the Explain tables. In addition, Explain information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to `NO`. For more information about special registers, see the *SQL Reference*.

**Note:** This value for EXPLAIN is not supported by DRDA.

**EXPLSNAP**
Stores Explain Snapshot information in the Explain tables. This DB2 precompile/bind option is not supported by DRDA.

**NO** An Explain Snapshot will not be captured.

**YES** An Explain Snapshot for each eligible static SQL statement will be placed in the Explain tables.

**ALL** An Explain Snapshot for each eligible static SQL statement will be placed in the Explain tables. In addition, Explain Snapshot information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to `NO`. For more information about special registers, see the *SQL Reference*.

**FUNCPATH**
Specifies the function path to be used in resolving user-defined distinct types and functions in static SQL. If this option is not specified, the default function path is ″SYSIBM″,″SYSFUN″,USER where USER is the value of the USER special register. This DB2 precompile/bind option is not supported by DRDA.

**schema-name**
A short SQL identifier, either ordinary or delimited, which identifies a schema that exists at the application server. No validation that the schema exists is made at precompile or at bind time. The same schema cannot appear more than once in the function path. The number of schemas that can be specified is limited by the length of the resulting function path, which cannot exceed 254 bytes. The schema SYSIBM does not need to be explicitly specified; it is implicitly assumed to be the first schema if it is not included in the function path. For more information, see the *SQL Reference*.

**INSERT**
Allows a program being precompiled or bound against a DB2 Universal Database Extended Enterprise Edition server to request that data inserts be buffered to increase performance.

**BUF** Specifies that inserts from an application should be buffered.

**DEF** Specifies that inserts from an application should not be buffered.

**GENERIC string**
Provides a means of passing new bind options to a target DRDA database. Supports the binding of new options that are defined in the target database, but that are not known to the local command. Do not use this option to pass bind options that *are* defined in “BIND” on page 131 or “PRECOMPILE PROGRAM” on page 372. This option can substantially improve dynamic SQL performance. The syntax is as follows:

```
generic "option1 value1 option2 value2 ..."
```

Each option and value must be separated by one or more blank spaces. For example, if the target DRDA database is DB2 MVS Version 5, one could use:

```
generic "keepdynamic yes"
```

to bind the new **keepdynamic** YES option, which is not defined locally on the PRECOMPILE PROGRAM or the BIND command.

The maximum length of the string is 1023 bytes. This DRDA bind option is currently only supported by DB2 MVS Version 5; it is not supported by DB2.

**ISOLATION**
Determines how far a program bound to this package can be isolated from the effect of other executing programs. For more information about isolation levels, see the *SQL Reference*.

**CS** Specifies Cursor Stability as the isolation level.

**NC** No Commit. Specifies that commitment control is not to be used. This isolation level is not supported by DB2.

**RR** Specifies Repeatable Read as the isolation level.

**RS** Specifies Read Stability as the isolation level. Read Stability ensures that the execution of SQL statements in the package is isolated from other application processes for rows read and changed by the application.

**UR** Specifies Uncommitted Read as the isolation level.

**LANGLEVEL**
Specifies the SQL rules that apply for both the syntax and the semantics for both static and dynamic SQL in the application. This option is not supported by DB2 Connect. For more information about this option, see the *Application Development Guide*.

**MIA** Select the ISO/ANS SQL92 rules as follows:

- To support error SQLCODE or SQLSTATE checking, an SQLCA must be declared in the application code.
- C null-terminated strings are padded with blanks and always include a null-terminating character, even if truncation occurs.
- The FOR UPDATE clause is optional for all columns to be updated in a positioned UPDATE.
- A searched UPDATE or DELETE requires SELECT privilege on the object table of the UPDATE or DELETE statement if a column of the object table is referenced in the search condition or on the right hand side of the assignment clause.
- A column function that can be resolved using an index (for example MIN or MAX) will also check for nulls and return warning SQLSTATE 01003 if there were any nulls.
- An error is returned when a duplicate unique constraint is included in a CREATE or ALTER TABLE statement.
- An error is returned when no privilege is granted and the grantor has no privileges on the object (otherwise a warning is returned).

**SAA1** Select the common IBM DB2 rules as follows:

- To support error SQLCODE or SQLSTATE checking, an SQLCA must be declared in the application code.
- C null-terminated strings are not terminated with a null character if truncation occurs.

- The FOR UPDATE clause is required for all columns to be updated in a positioned UPDATE.
- A searched UPDATE or DELETE will not require SELECT privilege on the object table of the UPDATE or DELETE statement unless a fullselect in the statement references the object table.
- A column function that can be resolved using an index (for example MIN or MAX) will not check for nulls and warning SQLSTATE 01003 is not returned.
- A warning is returned and the duplicate unique constraint is ignored.
- An error is returned when no privilege is granted.

**SQL92E**
Defines the ISO/ANS SQL92 rules as follows:

- To support checking of SQLCODE or SQLSTATE values, variables by this name may be declared in the host variable declare section (if neither is declared, SQLCODE is assumed during precompilation).
- C null-terminated strings are padded with blanks and always include a null-terminating character, even if truncation occurs.
- The FOR UPDATE clause is optional for all columns to be updated in a positioned UPDATE.
- A searched UPDATE or DELETE requires SELECT privilege on the object table of the UPDATE or DELETE statement if a column of the object table is referenced in the search condition or on the right hand side of the assignment clause.
- A column function that can be resolved using an index (for example MIN or MAX) will also check for nulls and return warning SQLSTATE 01003 if there were any nulls.
- An error is returned when a duplicate unique constraint is included in a CREATE or ALTER TABLE statement.
- An error is returned when no privilege is granted and the grantor has no privileges on the object (otherwise a warning is returned).

**LEVEL consistency-token**
Defines the level of a module using the consistency token. The consistency token is any alphanumeric value up to 8 characters in length. The RDB package consistency token verifies that the requester’s application and the relational database package are synchronized. This DRDA precompile option is not supported by DB2.

**Note:** This option is not recommended for general use.

**MESSAGES message-file**
Specifies the destination for warning, error, and completion status messages. A message file is created whether the bind is successful or not. If a message file name is not specified, the messages are written to standard output. If the complete path to the file is not specified, the current directory is used. If the name of an existing file is specified, the contents of the file are overwritten.

**NOLINEMACRO**
Suppresses the generation of the #line macros in the output `.c` file. Useful when the file is used with development tools which require source line information such as profiles, cross-reference utilities, and debuggers.

**Note:** This precompile option is used for the C/C$^{++}$ programming languages only.

**OPTLEVEL**
Indicates whether the C/C++ precompiler is to optimize initialization of internal SQLDAs when host variables are used in SQL statements. Such optimization can increase performance when a single SQL statement (such as FETCH) is used inside a tight loop.

**0** Instructs the precompiler not to optimize SQLDA initialization.

**1** Instructs the precompiler to optimize SQLDA initialization. This value should not be specified if the application uses:

- pointer host variables, as in the following example:

```
exec sql begin declare section;
char (*name)[20];
short *id;
exec sql end declare section;
```

- C++ data members directly in SQL statements.

For more information, see the *Application Development Guide*.

**OUTPUT filename**
Overrides the default name of the modified source file produced by the compiler. It can include a path.

**OWNER authorization-id**
Designates an 8-character authorization identifier for the package owner. The owner must have the privileges required to execute the SQL statements contained in the package. Only a user with SYSADM or DBADM authority can specify an authorization identifier other

than the user ID. The default value is the primary authorization ID of the precompile/bind process. SYSIBM, SYSCAT, and SYSSTAT are not valid values for this option.

**PACKAGE**
Creates a package. If neither **package**, **bindfile**, nor **syntax** is specified, a package is created in the database by default.

**USING package-name**
The name of the package that is to be generated by the precompiler. If a name is not entered, the name of the application program source file (minus extension and folded to uppercase) is used. Maximum length is 8 characters.

**PREPROCESSOR ″preprocessor-command″**

Specifies the preprocessor command that can be executed by the precompiler before it processes embedded SQL statements. The preprocessor command string (maximum length 1024 bytes) must be enclosed either by double or by single quotation marks.

This option enables the use of macros within the declare section. A valid preprocessor command is one that can be issued from the command line to invoke the preprocessor without specifying a source file. For example,

```
xlc -P -DMYMACRO=0
```

**QUALIFIER qualifier-name**
Provides an 8-character implicit qualifier for unqualified objects contained in the package. The default is the owner’s authorization ID, whether or not **owner** is explicitly specified.

**QUERYOPT optimization-level**
Indicates the desired level of optimization for all static SQL statements contained in the package. The default value is 5. For the complete range of optimization levels available, see the SET CURRENT QUERY OPTIMIZATION statement in the *SQL Reference*. This DB2 precompile/bind option is not supported by DRDA.

**RELEASE**
Indicates whether resources are released at each COMMIT point, or when the application terminates. This DRDA precompile/bind option is not supported by DB2.

**COMMIT**
Release resources at each COMMIT point. Used for dynamic SQL statements.

**DEALLOCATE**
Release resources only when the application terminates.

**SQLCA**

For FORTRAN applications only. This option is ignored if it is used with other languages.

**NONE**
: Specifies that the modified source code is not consistent with the SAA definition.

**SAA**
: Specifies that the modified source code is consistent with the SAA definition.

**SQLERROR**

Indicates whether to create a package or a bind file if an error is encountered.

**CHECK**
: Specifies that the target system performs all syntax and semantic checks on the SQL statements being bound. A package will not be created as part of this process. If, while binding, an existing package with the same name and version is encountered, the existing package is neither dropped nor replaced even if **action replace** was specified.

**CONTINUE**
: A package or a bind file is created even when SQL errors are encountered. This option is not supported by DB2.

**NOPACKAGE**
: A package or a bind file is not created if an error is encountered.

**SQLFLAG**

Identifies and reports on deviations from the SQL language syntax specified in this option.

A bind file or a package is created only if the **bindfile** or the **package** option is specified, in addition to the **sqlflag** option.

Local syntax checking is performed only if one of the following options is specified:

- **bindfile**
- **package**
- **sqlerror check**
- **syntax**

If **sqlflag** is not specified, the flagger function is not invoked, and the bind file or the package is not affected.

**SQL92E SYNTAX**
: The SQL statements will be checked against ANSI or ISO

SQL92 Entry level SQL language format and syntax with the exception of syntax rules that would require access to the database catalog. Any deviation is reported in the precompiler listing.

**MVSDB2V23 SYNTAX**
The SQL statements will be checked against MVS DB2 Version 2.3 SQL language syntax. Any deviation from the syntax is reported in the precompiler listing.

**MVSDB2V31 SYNTAX**
The SQL statements will be checked against MVS DB2 Version 3.1 SQL language syntax. Any deviation from the syntax is reported in the precompiler listing.

**MVSDB2V41 SYNTAX**
The SQL statements will be checked against MVS DB2 Version 4.1 SQL language syntax. Any deviation from the syntax is reported in the precompiler listing.

**SQLRULES**
Specifies:

- Whether type 2 CONNECTs are to be processed according to the DB2 rules or the Standard (STD) rules based on ISO/ANS SQL92.
- How a user or application can specify the format of LOB answer set columns.

**DB2**

- Permits the SQL CONNECT statement to switch the current connection to another established (*dormant*) connection.
- The user or application can specify the format of a LOB column only during the first fetch request.

**STD**

- Permits the SQL CONNECT statement to establish a *new* connection only. The SQL SET CONNECTION statement must be used to switch to a dormant connection.
- The user or application can change the format of a LOB column with each fetch request.

**SQLWARN**
Indicates whether warnings will be returned from the compilation of dynamic SQL statements (via PREPARE or EXECUTE IMMEDIATE), or from describe processing (via PREPARE...INTO or DESCRIBE). This DB2 precompile/bind option is not supported by DRDA.

| | |
|---|---|
| **NO** | Warnings will not be returned from the SQL compiler. |
| **YES** | Warnings will be returned from the SQL compiler. |

**Note:** SQLCODE +238 is an exception. It is returned regardless of the **sqlwarn** option value.

**STRDEL**

Designates whether an apostrophe (’) or double quotation marks (") will be used as the string delimiter within SQL statements. This DRDA precompile / bind option is not supported by DB2. The DRDA server will use a system defined default value if this option is not specified.

**APOSTROPHE**

Use an apostrophe (’) as the string delimiter.

**QUOTE**

Use double quotation marks (") as the string delimiter.

**SYNCPOINT**

Specifies how commits or rollbacks are to be coordinated among multiple database connections.

**NONE**

Specifies that no Transaction Manager (TM) is to be used to perform a two-phase commit, and does not enforce single updater, multiple reader. A COMMIT is sent to each participating database. The application is responsible for recovery if any of the commits fail.

**ONEPHASE**

Specifies that no TM is to be used to perform a two-phase commit. A one-phase commit is to be used to commit the work done by each database in multiple database transactions.

**TWOPHASE**

Specifies that the TM is required to coordinate two-phase commits among those databases that support this protocol.

**SYNTAX**

Suppresses the creation of a package or a bind file during precompilation. This option can be used to check the validity of the source file without modifying or altering existing packages or bind files. **Syntax** is a synonym for **sqlerror check**.

If **syntax** is used together with the **package** option, **package** is ignored.

**TARGET**

Instructs the precompiler to produce modified code tailored to one of the supported compilers on the current platform.

**IBMCOB**
On AIX, code is generated for the IBM COBOL Set for AIX compiler. On OS/2, code is generated for the IBM VisualAge for COBOL compiler.

**MFCOB**
Code is generated for the Micro Focus COBOL compiler. On OS/2 this refers to the 32-bit Micro Focus COBOL compiler. This is the default if a **target** value is not specified with the COBOL precompiler on all UNIX platforms and Windows NT.

**MFCOB16**
Code is generated for the 16-bit Micro Focus COBOL compiler. This value is only valid on OS/2, and is the default if a **target** value is not specified with the COBOL precompiler.

**ANSI_COBOL**
Code compatible with the ANS X3.23-1985 standard is generated.

**C** Code compatible with the C compilers supported by DB2 on the current platform is generated.

**CPLUSPLUS**
Code compatible with the C++ compilers supported by DB2 on the current platform is generated.

**BORLAND_C**
C code is generated for the Borland C/C++ compiler. This value is only valid on OS/2.

**BORLAND_CPLUSPLUS**
C++ code is generated for the Borland C/C++ compiler. This value is only valid on OS/2.

**FORTRAN**
Code compatible with the FORTRAN compilers supported by DB2 on the current platform is generated.

**TEXT label**
The description of a package. Maximum length is 255 characters. The default value is blanks. This DRDA precompile/bind option is not supported by DB2.

**VALIDATE**
Determines when the database manager checks for authorization errors and object not found errors. The package owner authorization ID is used for validity checking. This DRDA precompile/bind option is not supported by DB2.

**BIND** Validation is performed at precompile/bind time. If all objects do not exist, or all authority is not held, error messages are produced. If **sqlerror continue** is specified, a package / bind file is produced despite the error message, but the statements in error are not executable.

**RUN** Validation is attempted at bind time. If all objects exist, and all authority is held, no further checking is performed at execution time.

If all objects do not exist, or all authority is not held at precompile / bind time, warning messages are produced, and the package is successfully bound, regardless of the **sqlerror continue** option setting. However, authority checking and existence checking for SQL statements that failed these checks during the precompile / bind process may be redone at execution time.

**VERSION version-id**
Defines the version identifier for a package. The version identifier is any alphanumeric value, $, #, @, _, -, or ., up to 254 characters in length. This DRDA precompile option is not supported by DB2.

**WCHARTYPE**
For details and restrictions on the use and applicability of **wchartype**, see the *Application Development Guide*.

**CONVERT**
Host variables declared using the wchar_t base type will be treated as containing data in wchar_t format. Since this format is not directly compatible with the format of graphic data stored in the database (DBCS format), input data in wchar_t host variables is implicitly converted to DBCS format on behalf of the application, using the ANSI C function `wcstombs()`. Similarly, output DBCS data is implicitly converted to wchar_t format, using `mbstowcs()`, before being stored in host variables.

**NOCONVERT**
Host variables declared using the wchar_t base type will be treated as containing data in DBCS format. This is the format used within the database for graphic data; it is, however, different from the native wchar_t format implemented in the C language. Using **noconvert** means that graphic data will not undergo conversion between the application and the database, which can improve efficiency. The application is, however, responsible for ensuring that data in wchar_t format is not passed to the database manager. When this option is used, wchar_t host variables should not be manipulated with the C

wide character string functions, and should not be initialized with wide character literals (*L-literals*).

## Usage Notes

A modified source file is produced, which contains host language equivalents to the SQL statements. By default, a package is created in the database to which a connection has been established. The name of the package is the same as the file name (minus the extension and folded to uppercase), up to a maximum of 8 characters.

Following connection to a database, PREP executes under the transaction that was started. PREP then issues a COMMIT or a ROLLBACK to terminate the current transaction and start another one.

Creating a package with a schema name that does not already exist results in the implicit creation of that schema. The schema owner is SYSIBM. The CREATEIN privilege on the schema is granted to PUBLIC.

During precompilation, an Explain Snapshot is not taken unless a package is created and **explsnap** has been specified. The snapshot is put into the Explain tables of the user creating the package. Similarly, Explain table information is only captured when **explain** is specified, and a package is created.

Precompiling stops if a fatal error or more than 100 errors occur. If a fatal error occurs, the utility stops precompiling, attempts to close all files, and discards the package.

If a package is bound with **dynamicrules bind**, the implicit or explicit value of the bind option **owner** is used for authorization checking of dynamic SQL statements, and the implicit or explicit value of the bind option **qualifier** is used as the implicit qualifier of unqualified objects within dynamic SQL statements. If multiple packages are referenced during a single connection, dynamic SQL statements prepared by a specific package will behave according to the bind options for that package. The value of the special register CURRENT SCHEMA has no effect on qualification in a package bound with **dynamicrules bind**.

## See Also

"BIND" on page 131.

## PRUNE HISTORY/LOGFILE

Used to delete entries from the recovery history file, or to delete log files from the active log file path.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

### Required Connection

Database

### Command Syntax

```
►►──PRUNE───┬─HISTORY──timestamp──┬──────────────────┬─┬──────────────────────────►◄
            │                     └─WITH FORCE OPTION─┘ │
            └─LOGFILE PRIOR TO──log-file-name──────────┘
```

### Command Parameters

**HISTORY timestamp**
: Identifies a range of entries in the recovery history file that will be deleted. A complete time stamp (in the form *yyyymmddhhnnss*), or an initial prefix (minimum *yyyy*) can be specified. All entries with time stamps equal to or less than the time stamp provided are deleted from the recovery history file.

**WITH FORCE OPTION**
: Specifies that the entries will be pruned according to the time stamp specified, even if some entries from the most recent restore set are deleted from the file.

**LOGFILE PRIOR TO log-file-name**
: Specifies a string for a log file name, for example `S0000100.LOG`. The name identifies the log file that is to be deleted.

### Examples

To remove the entries for all restores, loads, table space backups, and full database backups taken before and including December 1, 1994 from the recovery history file, enter:

```
db2 prune history 199412
```

### Usage Notes

Pruning backup entries from the history file causes related file backups on DB2 Data Links Manager servers to be deleted.

## QUERY CLIENT

Returns current connection settings for an application process.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──QUERY CLIENT──────────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Examples

The following is sample output from QUERY CLIENT:

```
The current connection settings of the application process are:

                   CONNECT   = 1
                DISCONNECT   = EXPLICIT
   MAX_NETBIOS_CONNECTIONS   = 1
                  SQLRULES   = DB2
                 SYNCPOINT   = ONEPHASE
              CONNECT_NODE   = CATALOG_NODE
               ATTACH_NODE   = -1
```

If CONNECT_NODE and ATTACH_NODE are not set using the SET CLIENT command, these parameters have values identical to that of the environment variable DB2NODE. If the displayed value of the CONNECT_NODE or the ATTACH_NODE parameter is -1, the parameter has not been set; that is, either the environment variable DB2NODE has not been set, or the parameter was not specified in a previously issued SET CLIENT command.

### Usage Notes

The connection settings for an application process can be queried at any time during execution.

For information about distributed unit of work (DUOW), see the *Administration Guide*.

### See Also

“SET CLIENT” on page 465.

## QUIESCE TABLESPACES FOR TABLE

Quiesces table spaces for a table. There are three valid quiesce modes: share, intent to update, and exclusive. There are three possible states resulting from the quiesce function: QUIESCED SHARE, QUIESCED UPDATE, and QUIESCED EXCLUSIVE.

### Scope

In a single-partition environment, this command quiesces all table spaces involved in a load operation in exclusive mode for the duration of the load operation. In an MPP environment, this command acts locally on a node. It quiesces only that portion of table spaces belonging to the node on which the load operation is performed.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- *load*

### Required Connection

Database

### Command Syntax

```
►►──QUIESCE TABLESPACES FOR TABLE──┬─tablename────────┬──┬─SHARE───────────┬─────────►◄
                                   └─schema.tablename─┘  ├─INTENT TO UPDATE─┤
                                                         ├─EXCLUSIVE────────┤
                                                         └─RESET────────────┘
```

### Command Parameters

**TABLE**

**tablename**
: Specifies the unqualified table name. The table cannot be a system catalog table.

**schema.tablename**
: Specifies the qualified table name. If *schema* is not provided, the authorization ID used for the database connection will be used as the *schema*. The table cannot be a system catalog table.

**SHARE**
Specifies that the quiesce is to be in share mode.

When a ″quiesce share″ request is made, the transaction requests intent share locks for the table spaces and a share lock for the table. When the transaction obtains the locks, the state of the table spaces is changed to QUIESCED SHARE. The state is granted to the quiescer only if there is no conflicting state held by other users. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table, so that the state is persistent. The table cannot be changed while the table spaces for the table are in QUIESCED SHARE state. Other share mode requests to the table and table spaces are allowed. When the transaction commits or rolls back, the locks are released, but the table spaces for the table remain in QUIESCED SHARE state until the state is explicitly reset.

**INTENT TO UPDATE**
Specifies that the quiesce is to be in intent to update mode.

When a ″quiesce intent to update″ request is made, the table spaces are locked in intent exclusive (IX) mode, and the table is locked in update (U) mode. The state of the table spaces is recorded in the table space table.

**EXCLUSIVE**
Specifies that the quiesce is to be in exclusive mode.

When a ″quiesce exclusive″ request is made, the transaction requests super exclusive locks on the table spaces, and a super exclusive lock on the table. When the transaction obtains the locks, the state of the table spaces changes to QUIESCED EXCLUSIVE. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table. Since the table spaces are held in super exclusive mode, no other access to the table spaces is allowed. The user who invokes the quiesce function (the quiescer) has exclusive access to the table and the table spaces.

**RESET**
Specifies that the state of the table spaces is to be reset to normal.

## Examples

```
db2 quiesce tablespaces for table staff share
```

```
db2 quiesce tablespaces for table boss.org intent to update
```

## Usage Notes

A quiesce is a persistent lock. Its benefit is that it persists across transaction failures, connection failures, and even across system failures (such as power failure, or reboot).

A quiesce is owned by a connection. If the connection is lost, the quiesce remains, but it has no owner, and is called a *phantom quiesce*. A phantom quiesce becomes ″owned″ by the next connection that issues the QUIESCE TABLESPACES FOR TABLE command against the same table spaces or table. For example, if a power outage caused a load operation to be interrupted during the delete phase, the table spaces for the loaded table would be left in delete pending, quiesce exclusive state. Upon database restart, this quiesce would be an unowned (or phantom) quiesce.

To remove a phantom quiesce:

1. Connect to the database.
2. Use the LIST TABLESPACES command to determine which table space is quiesced.
3. Re-quiesce the table space using the current quiesce state. For example:

```
db2 quiesce tablespaces for table mytable exclusive
```

Once completed, the new connection owns the quiesce, and the load operation can be restarted.

There is a limit of five quiescers on a table space at any given time.

A quiescer can upgrade the state of a table space from a less restrictive state to a more restrictive one (for example, S to U, or U to X). If a user requests a state lower than one that is already held, the original state is returned. States are not downgraded.

## See Also

"LOAD" on page 338.

## QUIT

Exits the command line processor interactive input mode and returns to the operating system command prompt. If a batch file is being used to input commands to the command line processor, commands are processed until QUIT, TERMINATE, or the end-of-file is encountered.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──QUIT──────────────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Usage Notes

QUIT does not terminate the command line processor back-end process or break a database connection. CONNECT RESET breaks a connection, but does not terminate the back-end process. “TERMINATE” on page 484 does both.

## REBIND

Allows the user to recreate a package stored in the database without the need for a bind file.

### Authorization

One of the following:

- *sysadm* or *dbadm* authority
- ALTERIN privilege on the schema
- BIND privilege on the package.

The authorization ID logged in the BOUNDBY column of the SYSCAT.PACKAGES system catalog table, which is the ID of the most recent binder of the package, is used as the binder authorization ID for the rebind, and for the default *schema* for table references in the package. Note that this default qualifier may be different from the authorization ID of the user executing the rebind request. REBIND will use the same bind options that were specified when the package was created.

### Required Connection

Database. If no database connection exists, and if implicit connect is enabled, a connection to the default database is made.

### Command Syntax

```
►►──REBIND──┬─────────┬──package-name──────────────────────────────────►◄
            └─PACKAGE─┘
```

### Command Parameters

**PACKAGE package-name**
: The qualified or unqualified name that designates the package to be rebound. An unqualified package name is implicitly qualified by the current authorization ID.

### Usage Notes

REBIND does not automatically commit the transaction following a successful rebind. The user must explicitly commit the transaction. This enables ″what if″ analysis, in which the user updates certain statistics, and then tries to rebind the package to see what changes. It also permits multiple rebinds within a unit of work.

**Note:** The REBIND command *will* commit the transaction if auto-commit is enabled.

This command:

- Provides a quick way to recreate a package. This enables the user to take advantage of a change in the system without a need for the original bind file. For example, if it is likely that a particular SQL statement can take advantage of a newly created index, the REBIND command can be used to recreate the package. REBIND can also be used to recreate packages after “RUNSTATS” on page 461 has been executed, thereby taking advantage of the new statistics.
- Provides a method to recreate inoperative packages. Inoperative packages must be explicitly rebound by invoking either the bind utility or the rebind utility. A package will be marked inoperative (the VALID column of the `SYSCAT.PACKAGES` system catalog will be set to `X`) if a function instance on which the package depends is dropped.
- Gives users control over the rebinding of invalid packages. Invalid packages will be automatically (or implicitly) rebound by the database manager when they are executed. This may result in a noticeable delay in the execution of the first SQL request for the invalid package. It may be desirable to explicitly rebind invalid packages, rather than allow the system to automatically rebind them, in order to eliminate the initial delay and to prevent unexpected SQL error messages which may be returned in case the implicit rebind fails. For example, following migration, all packages stored in the database will be invalidated by the DB2 Version 5 migration process. Given that this may involve a large number of packages, it may be desirable to explicitly rebind all of the invalid packages at one time. This explicit rebinding can be accomplished using BIND, REBIND, or the **db2rbind** tool (see “db2rbind - Rebind all Packages” on page 69).

The choice of whether to use BIND or REBIND to explicitly rebind a package depends on the circumstances. It is recommended that REBIND be used whenever the situation does not specifically require the use of BIND, since the performance of REBIND is significantly better than that of BIND. BIND *must* be used, however:

- When there have been modifications to the program (for example, when SQL statements have been added or deleted, or when the package does not match the executable for the program).
- When the user wishes to modify any of the bind options as part of the rebind. REBIND does not support any bind options. For example, if the user wishes to have privileges on the package granted as part of the bind process, BIND must be used, since it has a **grant** option.
- When the package does not currently exist in the database.

- When detection of *all* bind errors is desired. REBIND only returns the first error it detects, whereas the BIND command returns the first 100 errors that occur during binding.

REBIND is supported by DB2 Connect.

If REBIND is executed on a package that is in use by another user, the rebind will not occur until the other user's logical unit of work ends, because an exclusive lock is held on the package's record in the `SYSCAT.PACKAGES` system catalog table during the rebind.

When REBIND is executed, the database manager recreates the package from the SQL statements stored in the `SYSCAT.STATEMENTS` system catalog table.

If REBIND encounters an error, processing stops, and an error message is returned.

REBIND will re-explain packages that were created with the **explsnap** bind option set to `YES` or `ALL` (indicated in the EXPLAIN_SNAPSHOT column in the `SYSCAT.PACKAGES` catalog table entry for the package) or with the **explain** bind option set to `YES` or `ALL` (indicated in the EXPLAIN_MODE column in the `SYSCAT.PACKAGES` catalog table entry for the package). The Explain tables used are those of the REBIND requester, not the original binder.

## See Also

"BIND" on page 131

"db2rbind - Rebind all Packages" on page 69

"RUNSTATS" on page 461.

## RECONCILE

Validates the references to files for the DATALINK data of a table. The rows for which the references to files cannot be established are copied to the exception table (if specified), and modified in the input table.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the table.

### Required Connection

Database

### Command Syntax

```
►►──RECONCILE──table-name──DLREPORT──filename──┬─────────────────────────────┬──────────►◄
                                               └─FOR EXCEPTION──table-name──┘
```

### Command Parameters

**RECONCILE table-name**
: Specifies the table on which reconciliation is to be performed. An alias, or the fully qualified or unqualified table name can be specified. A qualified table name is in the form *schema.tablename*. If an unqualified table name is specified, the table will be qualified with the current authorization ID.

**DLREPORT filename**
: Specifies the file that will contain information about the files that are unlinked during reconciliation. The name must be fully qualified (for example, `/u/johnh/report`). The reconcile utility appends a `.ulk` extension to the specified file name (for example, `report.ulk`).

**FOR EXCEPTION table-name**
: Specifies the exception table into which rows that encounter link failures for DATALINK values are to be copied.

### Examples

The following command reconciles the table DEPT, and writes exceptions to the exception table EXCPTAB, which was created by the user. Information about files that were unlinked during reconciliation is written into the file report.ulk, which is created in the directory /u/johnh. If FOR EXCEPTION excptab had not been specified, the exception information would have been written to the file report.exp, created in the /u/johnh directory.

```
   db2 reconcile dept dlreport /u/johnh/report for exception excptab
```

## Usage Notes

During reconciliation, attempts are made to link files which exist according to table data, but which do not exist according to Data Links File Manager metadata, if no other conflict exists.

Reconciliation is performed with respect to all DATALINK data in the table. If file references cannot be re-established, the violating rows are inserted into the exception table (if specified). These rows are not deleted from the input table. To ensure file reference integrity, the offending DATALINK values are nulled. If the column is defined as not nullable, the DATALINK values are replaced by a zero length URL.

If exception table is not specified, the DATALINK column values for which file references could not be re-established are copied to an exception report file (<*filename*>.exp), along with the column ID and a comment.

At the end of the reconciliation process, the table is taken out of datalink reconcile pending (DRP) state.

The exception table, if specified, should be created before the reconcile utility is run. The exception table used with the reconcile utility is identical to the exception table used by the load utility.

The exception table mimics the definition of the table being reconciled. It can have one or two optional columns following the data columns. The first optional column is the TIMESTAMP column. It will contain the time stamp for when the reconcile operation was started. The second optional column should be of type CLOB (32KB or larger). It will contain the IDs of columns with link failures, and the reasons for those failures. The DATALINK columns in the exception table should specify NO LINK CONTROL. This ensures that a file is not linked when a row (with a DATALINK column) is inserted, and that an access token is not generated when rows are selected from the exception table.

Information in the MESSAGE column is organized according to the following structure:

| Field number | Content | Size | Comments |
|---|---|---|---|
| 1 | Number of violations | 5 characters | Right justified padded with '0' |
| 2 | Type of violation | 1 character | 'L' - DATALINK violation |
| 3 | Length of violation | 5 characters | Right justified padded with '0' |
| 4 | Number of violating DATALINK columns | 4 characters | Right justified padded with '0' |
| 5 | DATALINK column number of the first violating column | 4 characters | Right justified padded with '0' |
| 6 | Reason for violation | 5 characters | Right justified padded with '0' |
| | | | Repeat Fields 5 and 6 for each violating column |

The following is a list of possible violations:

```
00001-File could not be found by DB2 Data Links Manager.
00002-File already linked.
00003-File in modified state.
00004-Prefix name not registered.
00005-File could not be retrieved.
00006-File entry missing. This will happen for
      recovery = no, partial control 1 DATALINK
      columns. Use update to relink the file.
00007-File is in unlink state.
00999-File could not be linked.

Example:

00001L000120002000400002000500001

   00001 - Specifies that the number of violations is 1.
   L     - Specifies that the type of violation is 'DATALINK violation'.
   00012 - Specifies that the length of the violation is 12 bytes.
   0002  - Specifies that there are 2 columns in the row which
           encountered link failures.
   0004,00002
   0005,00001 - Specifies the column ID and the reason for the violation.
```

If the message column is present, the time stamp column must also be present. For more information about exception tables, see the *SQL Reference*.

## REDISTRIBUTE NODEGROUP

Redistributes data across the nodes in a nodegroup. The current data distribution, whether it is uniform or skewed, can be specified. The redistribution algorithm selects the partitions to be moved based on the current data distribution.

This command can only be issued from the catalog node. Use "LIST DATABASE DIRECTORY" on page 299 to determine which node is the catalog node for each database.

### Scope

This command affects all nodes in the nodegroup.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *dbadm*

### Command Syntax

```
►►──REDISTRIBUTE NODEGROUP──nodegroup──┬─┬─UNIFORM──────────────┬─┬──────────────►◄
                                       │ └─USING DISTFILE──distfile─┘ │
                                       ├─USING TARGETMAP──targetmap───┤
                                       ├─CONTINUE─────────────────────┤
                                       └─ROLLBACK─────────────────────┘
```

### Command Parameters

**NODEGROUP nodegroup**
: The name of the nodegroup. This one-part name identifies a nodegroup described in the SYSNODEGROUPS catalog table. The nodegroup cannot currently be undergoing redistribution.

  **Note:** Tables in the IBMCATGROUP and the IBMTEMPGROUP nodegroups cannot be redistributed.

**UNIFORM**
: Specifies that the data is uniformly distributed across hash partitions (that is, every hash partition is assumed to have the same number of rows), but the same number of hash partitions do not map to each node. After redistribution, all nodes in the nodegroup have approximately the same number of hash partitions.

**USING DISTFILE distfile**

If the distribution of partitioning key values is skewed, use this option to achieve a uniform redistribution of data across the nodes of a nodegroup.

Use the *distfile* to indicate the current distribution of data across the 4 096 hash partitions.

Use row counts, byte volumes, or any other measure to indicate the amount of data represented by each hash partition. The utility reads the integer value associated with a partition as the weight of that partition. When a *distfile* is specified, the utility generates a target partitioning map that it uses to redistribute the data across the nodes in the nodegroup as uniformly as possible. After the redistribution, the weight of each node in the nodegroup is approximately the same (the weight of a node is the sum of the weights of all partitions that map to that node).

For example, the input distribution file may contain entries as follows:

```
10223
1345
112000
0
100
...
```

In the example, hash partition 2 has a weight of 112 000, and partition 3 (with a weight of 0) has no data mapping to it at all.

The *distfile* should contain 4 096 positive integer values in character format. The sum of the values should be less than or equal to 4 294 967 295.

If the path for *distfile* is not specified, the current directory is used.

**USING TARGETMAP targetmap**

The file specified in *targetmap* is used as the target partitioning map. Data redistribution is done according to this file. If the path is not specified, the current directory is used.

If a node included in the target map is not in the nodegroup, an error is returned. Issue ALTER NODEGROUP ADD NODE before running REDISTRIBUTE NODEGROUP.

If a node excluded from the target map *is* in the nodegroup, that node will not be included in the partitioning. Such a node can be dropped using ALTER NODEGROUP DROP NODE either before or after REDISTRIBUTE NODEGROUP.

**CONTINUE**
: Continues a previously failed REDISTRIBUTE NODEGROUP operation. If none occurred, an error is returned.

**ROLLBACK**
: Rolls back a previously failed REDISTRIBUTE NODEGROUP operation. If none occurred, an error is returned.

## Usage Notes

When a redistribution operation is done, a message file is written to:

- The `$HOME/sqllib/redist` directory on UNIX based systems, using the following format for subdirectories and file name: *database-name.nodegroup-name.timestamp*.
- The `$HOME\sqllib\redist\` directory on OS/2 or the Windows operating system, using the following format for subdirectories and file name: *database-name \ first-eight-characters-of-the-nodegroup-name \ date \ time*.

The time stamp value is the time when the command was issued.

This utility performs intermittent COMMITs during processing.

Use the ALTER NODEGROUP statement to add nodes to a nodegroup. This statement permits one to define the containers for the table spaces associated with the nodegroup. See the *SQL Reference* for details.

**Note:** DB2 Parallel Edition for AIX Version 1 syntax, with ADD NODE and DROP NODE options, is supported for users with *sysadm* or *sysctrl* authority. For ADD NODE, containers are created like the containers on the lowest node number of the existing nodes within the nodegroup.

All packages having a dependency on a table that has undergone redistribution are invalidated. It is recommended to explicitly rebind such packages after the redistribute nodegroup operation has completed. Explicit rebinding eliminates the initial delay in the execution of the first SQL request for the invalid package. The redistribute message file contains a list of all the tables that have undergone redistribution.

It is also recommended to update statistics by issuing "RUNSTATS" on page 461 after the redistribute nodegroup operation has completed.

Nodegroups containing replicated summary tables or tables defined with DATA CAPTURE CHANGES cannot be redistributed.

### See Also

"REBIND" on page 403.

## REFRESH LDAP

Refreshes the cache on a local machine with updated information when the information in LDAP (Lightweight Directory Access Protocol) has been changed.

This command is available on Windows NT, Windows 98, and Windows 95 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──REFRESH LDAP──┬─CLI CFG──┬───────────────────────────────────────────────►◄
                  ├─DB DIR───┤
                  └─NODE DIR─┘
```

### Command Parameters

**CLI CFG**
: Specifies that the CLI configuration is to be refreshed.

**DB DIR**
: Specifies that the database directory is to be refreshed.

**NODE DIR**
: Specifies that the node directory is to be refreshed.

### Usage Notes

If the object in LDAP is removed during refresh, the corresponding LDAP entry on the local machine is also removed. If the information in LDAP is changed, the corresponding LDAP entry is modified accordingly. If the `DB2CLI.INI` file is manually updated, the REFRESH LDAP CLI CFG command must be run to update the cache for the current user.

## REGISTER

Registers the DB2 server in the network directory server.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──REGISTER──┬───────────────────┬──┤ NWBINDERY path ├─┬──────────────────────►◄
              └─DB2 SERVER──IN────┘  └┤ LDAP path ├─────┘
```

#### NWBINDERY path:

```
├──NWBINDERY──USER──username──┬───────────────────────┬───────────────────────┤
                              └─PASSWORD──password────┘
```

#### LDAP path:

```
├──LDAP AS──nodename───────────────────────────────────────────────────────────►

►─PROTOCOL──┬─TCPIP──┬──────────────────────┬─┬──────────────────────┬─┬──────────────────┬─┬──►
            │        └─HOSTNAME──hostname───┘ └─SVCENAME──svcename───┘ └─SECURITY──SOCKS──┘ │
            ├─NETBIOS──┬───────────────┬──────────────────────────────────────────────────┤
            │          └─NNAME──nname──┘                                                    │
            ├─┤ APPN path ├───────────────────────────────────────────────────────────────┤
            ├─IPXSPX──┬────────────────────────┬──────────────────────────────────────────┤
            │         └─IPX_ADDRESS──ipxaddr───┘                                            │
            └─NPIPE─────────────────────────────────────────────────────────────────────────┘

►──┬──────────────────────┬──┬──────────────────────┬──┬──────────────────────────┬──────────►
   └─REMOTE──computer─────┘  └─INSTANCE──instance───┘  └─NODETYPE──┬─SERVER─┬─────┘
                                                                   ├─MPP────┤
                                                                   └─DCS────┘

►──┬────────────────────────┬──┬──────────────────────────────────────────────┬─────────────┤
   └─WITH──"comments"───────┘  └─USER──username──┬───────────────────────┬───┘
                                                  └─PASSWORD──password────┘
```

#### APPN path:

```
├──APPN──NETWORK──net_id──PARTNERLU──partner_lu──┬──────────────────────┬──MODE──mode──────►
                                                 └─TPNAME──tp_name──────┘
```

```
►─┬──────────────────────────────┬─┬────────────────────────────┬──────────────────────────►
  └─SECURITY──┬─NONE────┬────────┘ └─LANADDRESS──lan_address─┘
              ├─SAME────┤
              └─PROGRAM─┘

►─┬───────────────────────────────────┬──────────────────────────────────────────────────┤
  └─CHGPWDLU──change_password_lu──────┘
```

## Command Parameters

**IN** Specifies the network directory server on which to register the DB2 server. Valid values are: `NWBINDERY` for a NetWare bindery, and `LDAP` for an LDAP (Lightweight Directory Access Protocol) directory server.

**USER username**
: For NWBINDERY, this is the user ID to log into the network server. The user ID must have SUPERVISOR or Workgroup Manager security equivalence. The user name must be provided when registering in the NetWare directory server, and is the user ID to log into the NETWARE server. For LDAP, this is the user's LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create and update the object in the LDAP directory. The user name is optional when registering in LDAP. If the user's LDAP DN is not specified, the credentials of the current logon user will be used.

**PASSWORD password**
: Account password.

**AS nodename**
: Specify a short name to represent the DB2 server in LDAP. A node entry will be cataloged in LDAP using this node name. The client can attach to the server using this node name. The protocol associated with this LDAP node entry is specified through the PROTOCOL parameter.

**PROTOCOL**
: Specifies the protocol type associated with the LDAP node entry. Since the database server may support more than one protocol type, this value specifies the protocol type used by the client applications. The DB2 server must be registered once per protocol. Valid values are: `TCPIP`, `NETBIOS`, `APPN`, `IPXSPX`, and `NPIPE`. Specify the latter to use Windows NT Named Pipe. This protocol type is only supported by DB2 servers that run on the Windows NT operating system.

**HOSTNAME hostname**
: Specifies the TCP/IP host name (or IP address).

**SVCENAME svcename**
Specifies the TCP/IP service name or port number.

**SECURITY SOCKS**
Specifies that TCP/IP socket security is to be used.

**NNAME nname**
Specifies the NetBIOS workstation name.

**NETWORK net_id**
Specifies the APPN network ID.

**PARTNERLU partner_lu**
Specifies the APPN partner LU name for the DB2 server machine.

**TPNAME tpname**
Specifies the APPN transaction program name.

**MODE mode**
Specifies the APPN mode name.

**SECURITY**
Specifies the APPN security level. Valid values are:

**NONE**
Specifies that no security information is to be included in the allocation request sent to the server. This is the default security type for DB2 UDB server.

**SAME** Specifies that a user name is to be included in the allocation request sent to the server, together with an indicator that the user name has been ″already verified″. The server must be configured to accept ″already verified″ security.

**PROGRAM**
Specifies that both a user name and a password are to be included in the allocation request sent to the server. This is the default security type for host database servers such as DB2 for MVS, DB2 for AS/400, or DB2 for VM.

**LANADDRESS lan_address**
Specifies the APPN network adaptor address.

**CHGPWDLU change_password_lu**
Specifies the name of the partner LU that is to be used when changing the password for a host database server.

**IPX_ADDRESS ipxaddr**
Specifies the complete IPX address. The IPX address of a system on which DB2 UDB is installed can be found by invoking the **db2ipxad** command. The IPX address consists of a 12-digit network address, an

8-digit node address, and a 4-digit socket number: *<NetworkAddress>.<NodeAddress>.<socket>*

**REMOTE computer**
Specifies the computer name of the machine on which the DB2 server resides. Specify this parameter only if registering a remote DB2 server in LDAP. The value must be the same as the value specified when adding the server machine to LDAP. For the Windows NT operating system, this is the NT computer name. For UNIX based systems, this is the TCP/IP host name. For OS/2, this is the value specified for the **DB2SYSTEM** registry variable.

**INSTANCE instance**
Specifies the instance name of the DB2 server. The instance name must be specified for a remote instance (that is, when a value for the REMOTE parameter has been specified).

**NODETYPE**
Specifies the node type for the database server. Valid values are:

**SERVER**
Specify the SERVER node type for a DB2 UDB Enterprise Edition server. This is the default.

**MPP** Specify the MPP node type for a DB2 UDB Enterprise Edition - Extended (partitioned database) server.

**DCS** Specify the DCS node type when registering a host database server; this directs the client or gateway to use DRDA as the database protocol.

**WITH ″comments″**
Describes the DB2 server. Any comment that helps to describe the server registered in the network directory can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

## Usage Notes

Registering a DB2 server in NWBINDERY is only relevant when using file server addressing to connect a client and a server.

Register the DB2 server once for each protocol that the server supports. For example, if the DB2 server supports both NetBIOS and TCP/IP, the REGISTER command must be invoked twice:

```
db2 register db2 server in ldap as tcpnode protocol tcpip
db2 register db2 server in ldap as nbnode protocol netbios
```

The REGISTER command should be issued once for each DB2 server instance to publish the server in the directory server. If the communication parameter fields are reconfigured, or the server network address changes, update the DB2 server on the network directory server.

To update the DB2 server in LDAP, use the UPDATE LDAP NODE command after the changes have been made.

To update the DB2 server in NWBINDERY, deregister the DB2 server before making the changes, then register it again after the changes have been made.

If any protocol configuration parameter is specified when registering a DB2 server locally, it will override the value specified in the database manager configuration file.

For APPN, only the TPNAME is found in the database manager configuration file. To register APPN properly, values for the following mandatory parameters must be specified: NETID, PARTNERLU, TPNAME, MODE, and SECURITY. Values for the following optional parameters can also be specified: LANADDRESS and CHGPWDLU.

The REGISTER command can only be used to register a remote DB2 server in LDAP. The computer name and the instance name of the remote server must be specified, along with the communication protocol for the remote server.

When registering a host database server, a value of DCS must be specified for the NODETYPE parameter.

## See Also

"db2ipxad - Get IPX/SPX Internetwork Address" on page 48

"DEREGISTER" on page 196

"UPDATE LDAP NODE" on page 505.

## REORGANIZE TABLE

Reorganizes a table by reconstructing the rows to eliminate fragmented data, and by compacting information.

### Scope

This command affects all nodes in the nodegroup.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the table.

### Required Connection

Database

### Command Syntax

```
►►──REORG TABLE──table-name──┬────────────────────┬──┬──────────────────────────┬──────────►◄
                             └─INDEX──index-name──┘  └─USE──tablespace-name──┘
```

### Command Parameters

**TABLE table-name**
: Specifies the table to reorganize. The table can be in a local or a remote database. The fully qualified name or alias in the form: *schema.table-name* must be used. The *schema* is the user name under which the table was created.

**INDEX index-name**
: Specifies the index to use when reorganizing the table. The fully qualified name in the form: *schema.index-name* must be used. The *schema* is the user name under which the index was created. The database manager uses the index to physically reorder the records in the table it is reorganizing. If the name of an index is not provided, the records are reorganized without regard to order.

**USE tablespace-name**
: Specifies the name of a temporary table space where the database manager can temporarily store the table being reconstructed. If a table

space name is not entered, the database manager stores a working copy of the table in the table space(s) in which the table being reorganized resides.

For an 8KB table object, the page size of any temporary table space explicitly specified by the user must match the page size of the table space(s) in which the table data (including any LONG or LOB column data) resides.

## Examples

To reorganize the EMPLOYEE table using the temporary table space TEMPSPACE1 as a work area, enter:

```
db2 reorg table homer.employee using tempspace1
```

## Usage Notes

Tables that have been modified so many times that data is fragmented and access performance is noticeably slow are candidates for reorganization. Use “REORGCHK” on page 422 to determine whether a table needs reorganizing. Be sure to complete all database operations and release all locks before invoking REORGANIZE TABLE. This may be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK. After reorganizing a table, use “RUNSTATS” on page 461 to update the table statistics, and “REBIND” on page 403 to rebind the packages that use this table.

If the table is partitioned onto several nodes, and the table reorganization fails on any of the affected nodes, only the failing nodes will have the table reorganization rolled back.

**Note:** If the reorganization is not successful, temporary files should not be deleted. The database manager uses these files to recover the database.

If the name of an index is specified, the database manager reorganizes the data according to the order in the index. To maximize performance, specify an index that is often used in SQL queries. If the name of an index is *not* specified, and if a clustering index exists, the data will be ordered according to the clustering index.

The PCTFREE value of a table determines the amount of free space designated per page. If the value has not been set, the utility will fill up as much space as possible on each page.

This utility does not support the use of nicknames.

REORGANIZE TABLE cannot be used on views.

REORGANIZE TABLE cannot be used on a DMS table while an online backup of a table space in which the table resides is being performed.

To complete a table space roll-forward recovery following a table reorganization, both data and LONG table spaces must be roll-forward enabled.

If the table contains LOB columns that do not use the COMPACT option, the LOB DATA storage object can be significantly larger following table reorganization. This can be a result of the order in which the rows were reorganized, and the types of table spaces used (SMS/DMS).

DB2 Version 2 servers do not support down-level client requests to reorganize a table. Since pre-Version 2 servers do not support table spaces, the *tablespace-name* parameter is treated as the Version 1 *path* parameter, when Version 2 clients are used with a down-level server.

If a Version 2 client requests to reorganize a table on a Version 2 server, and that request includes a path instead of a temporary table space in the *tablespace-name* parameter (for example, an old application, specifying a temporary file path, being executed on Version 2 clients), REORG chooses a temporary table space in which to place the work files on behalf of the user. A valid temporary table space name containing a path separator character (/ or \) should not be specified, because it will be interpreted as a temporary path (pre-Version 2 request), and REORG will choose a temporary table space on behalf of the user.

## See Also

"REBIND" on page 403

"REORGCHK" on page 422

"RUNSTATS" on page 461.

## REORGCHK

Calculates statistics on the database to determine if tables need to be reorganized.

### Scope

This command can be issued from any node in the `db2nodes.cfg` file. It can be used to update table and index statistics in the catalogs.

### Authorization

One of the following:

- *sysadm* or *dbadm* authority
- CONTROL privilege on the table.

### Required Connection

Database

### Command Syntax

```
►►──REORGCHK──┬──────────────────────────────┬──┬─────────────────────────────┬──────────►◄
              │  ┌─UPDATE──┐                  │  │             ┌─USER───────┐  │
              └──┴─CURRENT─┴────STATISTICS────┘  └─ON TABLE────┼─SYSTEM─────┼──┘
                                                               ├─ALL────────┤
                                                               └─table-name─┘
```

### Command Parameters

**UPDATE STATISTICS**
: Calls the RUNSTATS routine to update table statistics, and then uses the updated statistics to determine if table reorganization is required.

  If a table partition exists on the node where REORGCHK has been issued, RUNSTATS executes on this node. If a table partition does not exist on this node, the request is sent to the first node in the nodegroup that holds a partition for the table. RUNSTATS then executes on that node.

**CURRENT STATISTICS**
: Uses the current table statistics to determine if table reorganization is required.

**ON TABLE**

  **USER** Checks the tables that are owned by the run time authorization ID.

**SYSTEM**
Checks the system tables.

**ALL** Checks all user and system tables.

**table-name**
Specifies the table to check. The fully qualified name or alias in the form: *schema.table-name* must be used. The *schema* is the user name under which the table was created. If the table specified is a system catalog table, the *schema* is SYSIBM.

## Examples

The following shows sample output from the command

```
db2 reorgchk update statistics on table system
```

run against the SAMPLE database:

```
Doing RUNSTATS ....

Table statistics:

F1: 100*OVERFLOW/CARD < 5
F2: 100*TSIZE / ((FPAGES-1) * (TABLEPAGESIZE-76)) > 70
F3: 100*NPAGES/FPAGES > 80

CREATOR  NAME                 CARD    OV    NP    FP    TSIZE  F1   F2  F3 REORG
--------------------------------------------------------------------------------
SYSIBM   SYSCHECKS               -     -     -     -        -   -    -   - ---
SYSIBM   SYSCOLAUTH              -     -     -     -        -   -    -   - ---
SYSIBM   SYSCOLCHECKS            -     -     -     -        -   -    -   - ---
SYSIBM   SYSCOLDIST              -     -     -     -        -   -    -   - ---
SYSIBM   SYSCOLUMNS            735     0    25    25    92610   0   95 100 ---
SYSIBM   SYSCONSTDEP             -     -     -     -        -   -    -   - ---
SYSIBM   SYSDATATYPES           13     0     1     1     1027   0    - 100 ---
SYSIBM   SYSDBAUTH               3     0     1     1       90   0    - 100 ---
SYSIBM   SYSEVENTMONITORS        -     -     -     -        -   -    -   - ---
SYSIBM   SYSEVENTS               -     -     -     -        -   -    -   - ---
SYSIBM   SYSFUNCPARMS          254     0     6     6    21590   0  100 100 ---
SYSIBM   SYSFUNCTIONS          104     0     8     8      728   0    2 100 -*-
SYSIBM   SYSINDEXAUTH            2     0     1     1      112   0    - 100 ---
SYSIBM   SYSINDEXES             57    17     3     5     9063  29   56  60 ***
SYSIBM   SYSKEYCOLUSE            4     0     1     1      268   0    - 100 ---
SYSIBM   SYSPLAN                22     0     2     2      154   0    3 100 -*-
SYSIBM   SYSPLANAUTH            41     0     1     1     1804   0    - 100 ---
SYSIBM   SYSPLANDEP              -     -     -     -        -   -    -   - ---
SYSIBM   SYSRELS                 -     -     -     -        -   -    -   - ---
SYSIBM   SYSSECTION              4     0     1     1      260   0    - 100 ---
SYSIBM   SYSSTMT                 4     0     1     1      268   0    - 100 ---
SYSIBM   SYSTABAUTH             68     0     2     2     3944   0   98 100 ---
SYSIBM   SYSTABCONST             2     0     1     1      132   0    - 100 ---
SYSIBM   SYSTABLES              69     0     6     6      483   0    2 100 -*-
SYSIBM   SYSTABLESPACES          3     0     1     1      225   0    - 100 ---
SYSIBM   SYSTRIGDEP             -     -     -     -        -   -   -   - ---
SYSIBM   SYSTRIGGERS            -     -     -     -        -   -   -   - ---
SYSIBM   SYSVIEWDEP            42     0     1     1     2646   0   - 100 ---
SYSIBM   SYSVIEWS              32     0     5     5     3168   0  19 100 -*-

Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100*(KEYS*(ISIZE+8)+(CARD-KEYS)*4) / (NLEAF*INDEXPAGESIZE) > 50
F6: (100-PCTFREE)*(INDEXPAGESIZE-96)/(ISIZE+12)**(NLEVELS-2))*(INDEXPAGESIZE-96)/
    (KEYS*(ISIZE+8)+(CARD-KEYS)*4) < 100

CREATOR  NAME                 CARD  LEAF  LVLS ISIZE   KEYS   F4   F5  F6 REORG
--------------------------------------------------------------------------------
Table: SYSIBM.SYSCHECKS
SYSIBM   IBM37                   -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSCOLAUTH
SYSIBM   IBM42                   -     -     -     -      -    -    -    - ---
SYSIBM   IBM43                   -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSCOLCHECKS
SYSIBM   IBM38                   -     -     -     -      -    -    -    - ---
SYSIBM   IBM39                   -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSCOLDIST
SYSIBM   IBM46                   -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSCOLUMNS
SYSIBM   IBM01                 735    12     2    33    735   97   64   11 ---
SYSIBM   IBM24                 735     1     1    20     10   85    -    - ---
Table: SYSIBM.SYSCONSTDEP
SYSIBM   IBM44                   -     -     -     -      -    -    -    - ---
SYSIBM   IBM45                   -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSDATATYPES
SYSIBM   IBM40                  13     1     1    20     13  100    -    - ---
SYSIBM   IBM41                  13     1     1     2     13  100    -    - ---
Table: SYSIBM.SYSDBAUTH
SYSIBM   IBM12                   3     1     1    17      3  100    -    - ---
Table: SYSIBM.SYSEVENTMONITORS
SYSIBM   IBM47                   -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSEVENTS
SYSIBM   IBM48                   -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSFUNCPARMS
SYSIBM   IBM31                 254     2     2    30    104  100   58   77 ---
SYSIBM   IBM32                 254     3     2    51    154   96   79   37 ---
SYSIBM   IBM33                 254     1     1     6      1  100    -    - ---
Table: SYSIBM.SYSFUNCTIONS
SYSIBM   IBM25                 104     1     1    30    104  100    -    - ---
SYSIBM   IBM26                 104     1     1    27    104   86    -    - ---
SYSIBM   IBM27                 104     1     1    18     50   86    -    - ---
SYSIBM   IBM28                 104     1     1    16      2   99    -    - ---
SYSIBM   IBM29                 104     1     1     4    104  100    -    - ---
SYSIBM   IBM30                 104     2     2    53    104   86   79   56 ---
Table: SYSIBM.SYSINDEXAUTH
SYSIBM   IBM17                   2     1     1    47      2  100    -    - ---
SYSIBM   IBM18                   2     1     1    30      2  100    -    - ---
Table: SYSIBM.SYSINDEXES
SYSIBM   IBM02                  57     1     1    17     57  100    -    - ---
SYSIBM   IBM03                 57     1     1    25     57  100    -    - ---
Table: SYSIBM.SYSKEYCOLUSE
SYSIBM   IBM35                  4     1     1    57      4  100    -    - ---
SYSIBM   IBM36                  4     1     1    44      2  100    -    - ---
Table: SYSIBM.SYSPLAN
SYSIBM   IBM07                 22     1     1    16     22  100    -    - ---
SYSIBM   IBM19                 22     1     1     8      1  100    -    - ---
Table: SYSIBM.SYSPLANAUTH
SYSIBM   IBM13                 41     1     1    33     41  100    -    - ---
SYSIBM   IBM14                 41     1     1    16     22  100    -    - ---
Table: SYSIBM.SYSPLANDEP
SYSIBM   IBM08                  -     -     -     -      -    -    -    - ---
SYSIBM   IBM09                  -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSRELS
SYSIBM   IBM20                  -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSSECTION
SYSIBM   IBM10                  4     1     1    20      4  100    -    - ---
Table: SYSIBM.SYSSTMT
SYSIBM   IBM11                  4     1     1    20      4  100    -    - ---
Table: SYSIBM.SYSTABAUTH
SYSIBM   IBM15                 68     1     1    38     68  100    -    - ---
SYSIBM   IBM16                 68     1     1    21     68  100    -    - ---
Table: SYSIBM.SYSTABCONST
SYSIBM   IBM34                  2     1     1    44      2  100    -    - ---
Table: SYSIBM.SYSTABLES
SYSIBM   IBM00                 69     1     1    21     69   95    -    - ---
SYSIBM   IBM21                 69     1     1    12      3  100    -    - ---
SYSIBM   IBM22                 69     1     1     6      1  100    -    - ---
SYSIBM   IBM23                 69     1     1     6      1  100    -    - ---
Table: SYSIBM.SYSTABLESPACES
SYSIBM   IBM49                  3     1     1    14      3  100    -    - ---
SYSIBM   IBM50                  3     1     1     8      1  100    -    - ---
Table: SYSIBM.SYSTRIGDEP
SYSIBM   IBM51                  -     -     -     -      -    -    -    - ---
SYSIBM   IBM52                  -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSTRIGGERS
SYSIBM   IBM53                  -     -     -     -      -    -    -    - ---
SYSIBM   IBM54                  -     -     -     -      -    -    -    - ---
Table: SYSIBM.SYSVIEWDEP
SYSIBM   IBM05                 42     1     1    42     42  100    -    - ---
SYSIBM   IBM06                 42     1     1    20     32  100    -    - ---
Table: SYSIBM.SYSVIEWS
SYSIBM   IBM04                 32     1     1    20     32  100    -    - ---
-------------------------------------------------------------------------------

CLUSTERRATIO or normalized CLUSTERFACTOR (F4) will indicate REORG is necessary
for indexes that are not in the same sequence as the base table. When multiple
indexes are defined on a table, one or more indexes may be flagged as needing
REORG.  Specify the most important index for REORG sequencing.
```

The terms for the table statistics (formulas 1-3) mean:

**CARD**
: Number of rows in base table.

**OV** (OVERFLOW) Number of overflow rows.

**NP** (NPAGES) Number of pages that contain data.

**FP** (FPAGES) Total number of pages.

**TSIZE** Table size in bytes. Calculated as the product of the number of rows in the table (CARD) and the average row length. The average row length is computed as the sum of the average column lengths (AVGCOLLEN in SYSCOLUMNS) plus 10 bytes of row overhead. For long fields and LOBs only the approximate length of the descriptor is used. The actual long field or LOB data is not counted in TSIZE.

**TABLEPAGESIZE**
Page size of the table space in which the table data resides.

**F1** Results of Formula 1.

**F2** Results of Formula 2.

**F3** Results of Formula 3.

**REORG**
Each hyphen (-) displayed in this column indicates that the calculated results were within the set bounds of the corresponding formula, and each asterisk (*) indicates that the calculated results exceeded the set bounds of its corresponding formula.

- - or * on the left side of the column corresponds to F1 (Formula 1)
- - or * in the middle of the column corresponds to F2 (Formula 2)
- - or * on the right side of the column corresponds to F3 (Formula 3).

Table reorganization is suggested when the results of the calculations exceed the bounds set by the formula.

For example, --- indicates that, since the formula results of F1, F2, and F3 are within the set bounds of the formula, no table reorganization is suggested. The notation *-* indicates that the results of F1 and F3 suggest table reorganization, even though F2 is still within its set bounds. The notation *-- indicates that F1 is the only formula exceeding its bounds.

**Note:** The table name is truncated to 30 characters, and the ″>″ symbol in the thirty-first column represents the truncated portion of the table name.

The terms for the index statistics (formulas 4-6) mean:

**CARD**
Number of rows in base table.

**LEAF** Total number of index leafs (pages).

**LVLS** (LEVELS) Number of index levels.

**ISIZE** Index size, calculated from the average column length of all columns participating in the index.

**KEYS** (FULLKEYCARD) Number of unique index entries.

**INDEXPAGESIZE**
Page size of the table space in which the table indexes reside, specified at the time of table creation. If not specified, INDEXPAGESIZE has the same value as TABLEPAGESIZE.

**PCTFREE**
Specifies the percentage of each index page to leave as free space, a value that is assigned when defining the index. Values can range from 0 to 99. The default value is 10.

**F4** Results of Formula 4.

**F5** Results of Formula 5. The notation +++ indicates that the result exceeds 999, and is invalid. Rerun REORGCHK with the UPDATE STATISTICS option, or issue "RUNSTATS" on page 461, followed by the REORGCHK command.

**F6** Results of Formula 6. The notation +++ indicates that the result exceeds 999, and is invalid. Rerun REORGCHK with the UPDATE STATISTICS option, or issue "RUNSTATS" on page 461, followed by the REORGCHK command.

**REORG**
Each hyphen (-) displayed in this column indicates that the calculated results were within the set bounds of the corresponding formula, and each asterisk (*) indicates that the calculated result exceeded the set bounds of its corresponding formula.

- `-` or `*` on the left side of the column corresponds to F4 (Formula 4)
- `-` or `*` in the middle of the column corresponds to F5 (Formula 5)
- `-` or `*` on the right side of the column corresponds to F6 (Formula 6).

Table reorganization is suggested when the results of the calculations exceed the bounds set by the formula.

## Usage Notes

This utility does not support the use of nicknames.

## REORGCHK

REORGCHK calculates statistics obtained from six different formulas to determine if performance has deteriorated or can be improved by reorganizing a table.

**Attention:** These statistics should not be used to determine if empty tables (TSIZE=0) need reorganization. If TSIZE=0 and FPAGE>0, the table needs to be reorganized. If TSIZE=0 and FPAGE=0, no reorganization is necessary.

REORGCHK uses the following formulas to analyze the physical location of rows and the size of the table:

- Formula F1:
  ```
  100*OVERFLOW/CARD < 5
  ```
  The total number of overflow rows in the table should be less than 5 percent of the total number of rows. Overflow rows can be created when rows are updated and the new rows contain more bytes than the old ones (VARCHAR fields), or when columns are added to existing tables.
- Formula F2:
  ```
  100*TSIZE / ((FPAGES-1) * (TABLEPAGESIZE-76)) > 70
  ```
  The table size in bytes (TSIZE) should be more than 70 percent of the total space allocated for the table. (There should be less than 30% free space.) The total space allocated for the table depends upon the page size of the table space in which the table resides (minus an overhead of 76 bytes). Because the last page allocated is not usually filled, 1 is subtracted from FPAGES.
- Formula F3:
  ```
  100*NPAGES/FPAGES > 80
  ```
  The number of pages that contain no rows at all should be less than 20 percent of the total number of pages. (Pages can become empty after rows are deleted.)

REORGCHK uses the following formulas to analyze the relationship of the indexes to the table data:

- Formula F4:
  ```
  CLUSTERRATIO or normalized CLUSTERFACTOR > 80
  ```
  The clustering ratio of an index should be greater than 80 percent. When multiple indexes are defined on one table, some of these indexes have a low cluster ratio. (The index sequence is not the same as the table sequence.) This cannot be avoided. Be sure to specify the most important index when reorganizing the table. The cluster ratio is usually not optimal for indexes that contain many duplicate keys and many entries.

- Formula F5:

  ```
  100*(KEYS*(ISIZE+8)+(CARD-KEYS)*4) / (NLEAF*INDEXPAGESIZE) > 50
  ```

  Less than 50 percent of the space reserved for index entries should be empty (only checked when NLEAF>1).
- Formula F6:

  ```
  (100-PCTFREE)*(INDEXPAGESIZE-96)/(ISIZE+12)**(NLEVELS-2))*(INDEXPAGESIZE-96)/
  (KEYS*(ISIZE+8)+(CARD-KEYS)*4) < 100
  ```

  The actual number of index entries should be more than 90% (or 100-PCTFREE) of the number of entries an NLEVELS-1 index tree can handle (only checked if NLEVELS>1).

**Note:** Running statistics on many tables can take time, especially if the tables are large.

## See Also

"REORGANIZE TABLE" on page 419

"RUNSTATS" on page 461.

## RESET ADMIN CONFIGURATION

Resets the parameters in the database manager configuration file that are relevant to the DB2 Administration Server to the system defaults. The values are reset by node type, which is always a server with remote clients. The DB2 Administration Server is a special DB2 instance that enables remote administration of DB2 servers. The following database manager configuration parameters are reset:

- AGENT_STACK_SZ
- AUTHENTICATION
- DIAGLEVEL
- DIAGPATH
- DISCOVER
- DISCOVER_COMM
- FILESERVER
- IPX_SOCKET
- NNAME
- OBJECTNAME
- QUERY_HEAP_SZ
- SYSADM_GROUP
- SYSCTRL_GROUP
- SYSMAINT_GROUP
- TPNAME
- TRUST_ALLCLNTS
- TRUST_CLNTAUTH

**Note:** It is not recommended that the SVCENAME parameter, set by the installation program, be modified by the user. The administration server service name is set to use the DB2 registered TCP/IP port (523).

For more information about these parameters, see "GET DATABASE MANAGER CONFIGURATION" on page 236.

### Scope

This command resets the database manager configuration file, `$HOME/sqllib/db2systm`. It affects all nodes that are listed in the `$HOME/sqllib/db2nodes.cfg` file.

## Authorization

*sysadm*

## Required Connection

None or instance. An instance attachment is not required to perform local DBM configuration operations, but is required to perform remote DBM configuration operations. To reset the database manager configuration for a remote instance, it is necessary to first attach to that instance.

## Command Syntax

```
►►──RESET ADMIN───┬─CONFIGURATION─┬──────────────────────────────────────────────────►◄
                  ├─CONFIG────────┤
                  └─CFG───────────┘
```

## Command Parameters

None

## Usage Notes

To view or print a list of the admin configuration parameters, use "GET ADMIN CONFIGURATION" on page 217.

To change the value of an admin parameter, use "UPDATE ADMIN CONFIGURATION" on page 494.

For more information about these parameters, see the *Administration Guide*.

Changes to the database manager configuration file become effective only after they are loaded into memory. This occurs during execution of **db2start**.

If an error occurs, the database manager configuration file does not change.

The database manager configuration file cannot be reset if the checksum is invalid. This may occur if the database manager configuration file is changed without using the appropriate command. If this happens, the database manager must be installed again to reset the database manager configuration file.

## See Also

"GET ADMIN CONFIGURATION" on page 217

## RESET ADMIN CONFIGURATION

"UPDATE ADMIN CONFIGURATION" on page 494.

## RESET DATABASE CONFIGURATION

Resets the configuration of a specific database to the system defaults.

### Scope

This command only affects the node on which it is executed.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance. An explicit attachment is not required. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command.

### Command Syntax

```
►►──RESET───┬─DATABASE─┬───┬─CONFIGURATION─┬──FOR──database-alias──────────────────────►◄
            └─DB───────┘   ├─CONFIG────────┤
                           └─CFG───────────┘
```

### Command Parameters

**FOR database-alias**
Specifies the alias of the database whose configuration is to be reset to the system defaults.

### Usage Notes

To view or print a list of the database configuration parameters, use “GET DATABASE CONFIGURATION” on page 225.

To change the value of a configurable parameter, use “UPDATE DATABASE CONFIGURATION” on page 501.

For more information about these parameters, see the *Administration Guide*.

Changes to the database configuration file become effective only after they are loaded into memory. All applications must disconnect from the database before this can occur.

If an error occurs, the database configuration file does not change.

The database configuration file cannot be reset if the checksum is invalid. This may occur if the database configuration file is changed without using the appropriate command. If this happens, the database must be restored to reset the database configuration file.

## See Also

"GET DATABASE CONFIGURATION" on page 225

"UPDATE DATABASE CONFIGURATION" on page 501.

## RESET DATABASE MANAGER CONFIGURATION

Resets the parameters in the database manager configuration file to the system defaults. The values are reset by node type, which is always a server with remote clients.

### Authorization

*sysadm*

### Required Connection

None or instance. An instance attachment is not required to perform local DBM configuration operations, but is required to perform remote DBM configuration operations. To reset the database manager configuration for a remote instance, it is necessary to first attach to that instance.

### Command Syntax

```
►►──RESET───┬─DATABASE MANAGER─┬───┬─CONFIGURATION─┬──────────────────────────────────►◄
            ├─DB MANAGER───────┤   ├─CONFIG────────┤
            └─DBM──────────────┘   └─CFG───────────┘
```

### Command Parameters

None

### Usage Notes

To view or print a list of the database manager configuration parameters, use "GET DATABASE MANAGER CONFIGURATION" on page 236.

To change the value of a configurable parameter, use "UPDATE DATABASE MANAGER CONFIGURATION" on page 503.

For more information about these parameters, see the *Administration Guide*.

Changes to the database manager configuration file become effective only after they are loaded into memory. For a server configuration parameter, this occurs during execution of **db2start**. For a client configuration parameter, this occurs when the application is restarted. If the client is the command line processor, it is necessary to invoke "TERMINATE" on page 484.

If an error occurs, the database manager configuration file does not change.

The database manager configuration file cannot be reset if the checksum is invalid. This may occur if the database manager configuration file is changed without using the appropriate command. If this happens, the database manager must be installed again to reset the database manager configuration file.

## See Also

"GET DATABASE MANAGER CONFIGURATION" on page 236

"UPDATE DATABASE MANAGER CONFIGURATION" on page 503.

## RESET MONITOR

Resets the internal database system monitor data areas of a specified database, or of all active databases, to zero. The internal database system monitor data areas include the data areas for all applications connected to the database, as well as the data areas for the database itself.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance or database:
- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To reset the monitor switches for a remote instance (or a different local instance), it is necessary to first attach to that instance.

### Command Syntax

```
►►──RESET MONITOR──┬─ALL──┬─────┬─────────────────────────────────┬─────────────►◄
                   │      └─DCS─┘                                 │
                   └─FOR──┬─────┬──┬─DATABASE─┬──database-alias───┘
                          └─DCS─┘  └─DB───────┘
```

### Command Parameters

**ALL**
: This option indicates that the internal counters should be reset for all databases.

**FOR DATABASE database-alias**
: This option indicates that only the database with alias *database-alias* should have its internal counters reset.

**DCS**
: Depending on which clause it is specified, this keyword resets the internal counters of:
  - All DCS databases
  - A specific DCS database.

## Usage Notes

Each process (attachment) has its own private view of the monitor data. If one user resets, or turns off a monitor switch, other users are not affected. Change the setting of the monitor switch configuration parameters to make global changes to the monitor switches (see “UPDATE DATABASE MANAGER CONFIGURATION” on page 503).

If ALL is specified, some database manager information is also reset to maintain consistency of the returned data, and some node-level counters are reset.

To see a list of the data items that can be reset, see the *System Monitor Guide and Reference*.

## See Also

“GET SNAPSHOT” on page 254

“GET MONITOR SWITCHES” on page 252.

## RESTART DATABASE

Restarts a database that has been abnormally terminated and left in an inconsistent state. At the successful completion of RESTART DATABASE, the application remains connected to the database if the user has CONNECT privilege.

### Scope

This command affects only the node on which it is executed.

### Authorization

None

### Required Connection

This command establishes a database connection.

### Command Syntax

```
►►──RESTART──┬─DATABASE─┬──database-alias─────────────────────────────────────►
             └─DB───────┘

►─┬─────────────────────────────────────┬─────────────────────────────────────►
  └─USER──username──┬───────────────────┤
                    └─USING──password───┘

►─┬────────────────────────────────────────────────────────┬──────────────────►◄
  │                                ┌───────────────────┐   │
  └─DROP PENDING TABLESPACES──(────▼───tablespace-name──┴──)──┘
```

### Command Parameters

**DATABASE database-alias**
Identifies the database to restart.

**USER username**
Identifies the user name under which the database is to be restarted.

**USING password**
The password used to authenticate *username*. If the password is omitted, the user is prompted to enter it.

**DROP PENDING TABLESPACES tablespace-name**
Specifies that the database restart operation is to be successfully completed even if table space container problems are encountered.

If a problem occurs with a container for a specified table space during the restart process, the corresponding table space will not be available (it will be in drop pending state) after the restart operation. In the case of circular logging, a troubled table space will cause a restart failure. A list of troubled table space names can found in `db2diag.log` if a restart database operation fails because of container problems. If there is only one temporary table space in the database, and it is in drop pending state, a new temporary table space must be created immediately following a successful database restart operation.

## Usage Notes

Execute this command if an attempt to connect to a database returns an error message, indicating that the database must be restarted. This action occurs only if the previous session with this database terminated abnormally (due to power failure, for example).

At the completion of RESTART DATABASE, a shared connection to the database is maintained if the user has CONNECT privilege, and an SQL warning is issued if any indoubt transactions exist. In this case, the database is still usable, but if the indoubt transactions are not resolved before the last connection to the database is dropped, another RESTART DATABASE must be issued before the database can be used again. Use “LIST INDOUBT TRANSACTIONS” on page 314 to generate a list of indoubt transactions. For more information about indoubt transactions, see the *Administration Guide*.

If the database is only restarted on a single node within an MPP system, a message may be returned on a subsequent database query indicating that the database needs to be restarted. This occurs because the database partition on a node on which the query depends must also be restarted. Restarting the database on all nodes solves the problem.

## See Also

CONNECT TO statement in the *SQL Reference*.

## RESTORE DATABASE

Rebuilds a damaged or corrupted database that has been backed up using BACKUP DATABASE. The restored database is in the same state it was in when the backup copy was made. This utility can also restore to a database with a name different from the database name in the backup image (in addition to being able to restore to a new database).

The utility can also be used to restore previous versions of DB2 databases.

If, at the time of the backup operation, the database was enabled for roll-forward recovery, the database can be brought to the state it was in prior to the occurrence of the damage or corruption by issuing ROLLFORWARD DATABASE after successful execution of RESTORE DATABASE.

This utility can also restore from a table space level backup.

To restore a database that was backed up on a different workstation platform, use “db2move - Database Movement Tool” on page 56.

### Scope

This command only affects the node on which it is executed.

### Authorization

To restore to an existing database, one of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

To restore to a new database, one of the following:

- *sysadm*
- *sysctrl*

### Required Connection

Database, to restore to an existing database.

Instance and database, to restore to a new database. The instance attachment is required to create the database.

To restore to a new remote database, it is necessary to first attach to the instance where the new database will reside.

## RESTORE DATABASE

### Command Syntax

```
>>──RESTORE──┬─DATABASE─┬──source-database-alias──┬─┤ restore-options ├─┬──────────><
             └─DB───────┘                         ├─CONTINUE───────────┤
                                                  └─ABORT──────────────┘
```

**restore-options:**

```
├──┬──────────────────────────────────────────┬────────────────────────────────────>
   └─USER──username──┬──────────────────────┬─┘
                     └─USING──password──────┘

>──┬────────────────────────────────────────────────────────┬──────────────────────>
   ├─TABLESPACE ONLINE──────────────────────────────────────┤
   │                  ┌─,─────────────────┐                 │
   ├─TABLESPACE──(────▼──tablespace-name──┴──)──┬────────┬──┤
   │                                            └─ONLINE─┘  │
   └─HISTORY FILE──┬────────┬───────────────────────────────┘
                   └─ONLINE─┘

>──┬──────────────────────────────────────────────────────────┬────────────────────>
   ├─USE ADSM──┬───────────────────────────────────┬──────────┤
   │           └─OPEN──num-sessions──SESSIONS──────┘          │
   │        ┌─,─────────────┐                                 │
   ├─FROM───▼──┬─directory─┬┴─────────────────────────────────┤
   │           └─device────┘                                  │
   └─LOAD──shared-library──┬──────────────────────────────────┤
                           └─OPEN──num-sessions──SESSIONS─────┘

>──┬───────────────────────┬──┬────────────────────────┬───────────────────────────>
   └─TAKEN AT──date-time───┘  └─TO──target-directory───┘

>──┬───────────────────────────────┬──┬───────────────────────────┬────────────────>
   └─INTO──target-database-alias───┘  └─NEWLOGPATH──directory─────┘

>──┬──────────────────────────────────┬──┬───────────────────────────┬─────────────>
   └─WITH──num-buffers──BUFFERS───────┘  └─BUFFER──buffer-size───────┘

>──┬───────────────────────┬──┬──────────────────┬──┬───────────┬──┬─────────────────┬──>
   └─DLREPORT──filename────┘  └─REPLACE EXISTING─┘  └─REDIRECT──┘  └─PARALLELISM──n──┘
```

```
►─┬────────────────────────┬─┬───────────────────┬─┬─────────────────────┬──────────┤
  └─WITHOUT ROLLING FORWARD─┘ └─WITHOUT DATALINK─┘ └─WITHOUT PROMPTING─┘
```

## Command Parameters

**DATABASE source-database-alias**
Alias of the source database from which the backup was taken.

**CONTINUE**
Indicates that the containers have been redefined, and that the final step in the redirected restore should be performed.

**ABORT**
Stops the redirected restore. Useful when an error has occurred that would require one or more steps to be repeated. After RESTORE DATABASE with the ABORT option has been issued, each step of a redirected restore must be repeated, including RESTORE DATABASE with the REDIRECT option.

**USER username**
Identifies the user name under which the database is to be restored.

**USING password**
The password used to authenticate the user name. If the password is omitted, the user is prompted to enter it.

**TABLESPACE tablespace-name**
A list of names used to specify the table spaces that are to be restored.

**ONLINE**
This keyword, applicable only when doing a table space level restore, is specified to allow the backup to be restored online. This means that other agents can connect while the backup is being restored.

**HISTORY FILE**
This keyword is specified to restore the history file from the backup only.

**USE ADSM**
Indicates that the database is to be restored from ADSM-managed output.

**OPEN num-sessions SESSIONS**
The number of I/O sessions to be used with ADSM or the vendor product.

**FROM directory/device**
The directory or device on which the backup images reside. If USE ADSM, FROM, and LOAD are omitted, the default is the current directory.

**Note:** On OS/2 or the Windows operating system, the specified directory must not be a DB2-generated directory. For example, given the following commands:

```
db2 backup database sample to c:\backup
db2 restore database sample from c:\backup
```

DB2 generates subdirectories under the `c:\backup` directory that should be ignored. To specify precisely which backup image to restore, use the TAKEN AT parameter. There may be several backup images stored in the same path.

If several items are specified, and the last item is a tape device, the user is prompted for another tape. Valid response options are:

**c** Continue. Continue using the device that generated the warning message (for example, when a new tape has been mounted)

**d** Device terminate. Stop using **only** the device that generated the warning message (for example, when there are no more tapes)

**t** Terminate. Abort the restore or backup utility.

Tape is not supported on OS/2. On OS/2, `0` or `0:` can be specified to cause the restore operation to call the user exit program (see the *Administration Guide*). This option is not valid on any other platform.

**Note:** Redirected restore is not allowed when a user exit program is used to perform the restore.

**LOAD shared-library**
The name of the shared library (DLL on OS/2 or the Windows operating system) containing the vendor backup and restore I/O functions to be used. It may contain the full path. If the full path is not given, it will default to the path where the user exit programs reside.

**TAKEN AT date-time**
The time stamp of the database backup. The backup image file name includes the time stamp.

**TO target-directory**
Directory of the target database. This parameter is ignored if the utility is restoring to an existing database.

**INTO target-database-alias**
Alias of the target database. If the target database does not exist, it will be created.

**NEWLOGPATH directory**
A fully qualified directory where recovery log files for the database being restored will be kept. These logs will then be used during rollforward, and any subsequent operation requiring logging. This parameter has the same function as the database configuration parameter *newlogpath*, except that its effect is limited to the RESTORE command in which it is specified.

**WITH num-buffers BUFFERS**
The number of buffers to be used.

**BUFFER buffer-size**
The size, in pages, of the buffer used for the restore operation. The minimum value for this parameter is 16 pages; the default value is 1024 pages. If a buffer-size of zero is specified, the value in the database manager configuration parameter *restbufsz* must be set to 16.

The specified value is compared to the value specified during the backup operation. The actual restore buffer size will be an even multiple of the backup buffer size, which is equal to or greater than the backup buffer size. For example, if a backup buffer size of 1024 pages were specified, and an attempt were made to restore this backup with a buffer size of 16 pages, the actual restore buffer size would be 1024. If the specified restore buffer size were 2049, the actual restore buffer size would be 2048.

To use tape devices, DB2 users on SCO UnixWare 7 must specify a buffer size of 16.

**DLREPORT filename**
The file name, if specified, must be fully qualified. The files which become unlinked during restore (as a result of a fast reconcile) will be reported. This option is only to be used if the table being restored has a DATALINK column type and data linked files.

**REPLACE EXISTING**
If a database with the same alias as the target database alias already exists, this parameter tells the restore utility to replace the existing database with the restored database. This is useful in scripts containing the RESTORE DATABASE command, because the CLP will not prompt the user to verify deletion of the existing database. If the WITHOUT PROMPTING parameter is specified, it is not necessary to specify REPLACE EXISTING, but in this case the command will fail if events occur that normally require user intervention.

**REDIRECT**
Specifies a redirected restore. To complete a redirected restore, this command should be followed by one or more SET TABLESPACE

CONTAINERS commands, and then by a RESTORE DATABASE command with the CONTINUE option.

**Note:** All commands associated with a single redirected restore must be executed from the same window or CLP session.

**WITHOUT ROLLING FORWARD**
Specifies not to place the database in roll-forward pending state after it has been successfully restored.

If, following a successful restore, the database is in roll-forward pending state, ROLLFORWARD DATABASE must be executed before the database can be used.

**WITHOUT DATALINK**
Specifies that any tables with DATALINK columns be placed in DataLink_Reconcile_Pending (DRP) state, and that no reconciliation of linked files is to be performed.

**PARALLELISM n**
Specifies the number of buffer manipulators to be spawned during the restore process. The default value is 1.

**WITHOUT PROMPTING**
Specifies that the restore will run unattended, and that any actions which normally require user intervention will instead return an error message.

## Examples

Following is a typical redirected restore scenario for a database whose alias is MYDB:

1. Issue a RESTORE DATABASE command with the REDIRECT option.

   ```
   db2 restore db mydb replace existing redirect
   ```

   After successful completion of step 1, and before completing step 3, the restore can be aborted by issuing:

   ```
   db2 restore db mydb abort
   ```

2. Issue a SET TABLESPACE CONTAINERS command for each table space whose containers must be redefined. For example, on OS/2:

   ```
   db2 set tablespace containers for 5 using
      (file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
   ```

   To verify that the containers of the restored database are the ones specified in this step, issue the LIST TABLESPACE CONTAINERS command.

3. After successful completion of steps 1 and 2, issue:

   ```
   db2 restore db mydb continue
   ```

This is the final step of the redirected restore.

4. If step 3 fails, or if the restore has been aborted, the redirected restore can be restarted, beginning at step 1.

## Usage Notes

**Database Level Restore**

If restoring to an existing database, the current database configuration file is not replaced by the backup copy unless the configuration file is corrupt.

If WITHOUT ROLLING FORWARD is not specified, and the database was enabled for roll-forward recovery at the time it was backed up, the database is in roll-forward pending state after it has been successfully restored. Use “GET DATABASE CONFIGURATION” on page 225 to check the database state. If the database is in roll-forward pending state, “ROLLFORWARD DATABASE” on page 451 must be issued against the database before it can be used.

BACKUP and RESTORE can also be used to copy a database to another file system or node.

If the backup file being restored was created during an online backup, it is imperative that forward recovery be invoked at the completion of the restore. Forward recovery (using ROLLFORWARD DATABASE) will ensure that any changes which occurred during the course of the backup operation are captured to bring the database into a stable state.

For offline restore, this utility connects to the database in exclusive mode. The utility fails if any application, including the calling application, is already connected to the database that is being restored to.

If an interrupt occurs during a restore, it will not be possible to successfully connect to the database until a successful restore has been performed.

The backup image must be an image that was created by the BACKUP DATABASE command, and may reside on disk, diskette (on OS/2 or the Windows operating system), tape, at the ADSM utility, or on other vendor product-managed media. Tape is not supported on OS/2.

When a database backup is restored to an existing database, the database inherits the alias and database names of the existing database. When restoring to a nonexistent database, the new database will be created with an alias and database name specified by the *target-database-alias* parameter. If a target database alias is not specified, the database will inherit the alias and database name of the backed up database.

Although a remote client may initiate a restore, the source and target always refer to entities that exist at the server.

Restoring databases may have prerequisite requirements and restrictions that are beyond the scope of this manual. For more detailed information about these conditions, see the *Administration Guide*.

**Table Space Level Restore**

To ensure that restored table spaces are synchronized with the rest of the database, the table spaces must be rolled forward to the end of the log (or to the point where the table spaces were last used). For this reason, table space level backup and restore can only be performed if roll-forward recovery is enabled. If roll-forward recovery is disabled at any time after a table space level backup is executed, it will not be possible to restore from the backup, and then to roll the table space forward to the current point in time. In this case, all table space level backups taken prior to that time are no longer restorable. The restore operation will fail if the user tries to restore from such a backup. In cases where it cannot be determined that the backup is invalid (if, for instance, the database has been restored and rolled forward, thus creating a new log sequence), the restore may be successful, and the broken restore set will be detected during roll-forward recovery.

Each component of a table may be backed up and restored with the table space in which it resides, independently of the other components of the table.

Table space level backup and restore cannot be run concurrently.

**DB2 Data Links Manager Considerations**

When restoring to a database name or alias different from that of the backup image, tables with DATALINK columns are put in DRNP state.

If the WITHOUT DATALINK option is not specified, and the DB2 File Manager containing the DATALINK data is unavailable, the restore operation will fail. If this option is specified, and the DB2 File Manager containing the DATALINK data is unavailable, all table spaces which contain tables with DATALINK values on the unavailable server are placed in restore pending state.

For detailed information about DB2 Data Links Manager and database recovery, see the *Administration Guide*.

## See Also

"db2move - Database Movement Tool" on page 56

"BACKUP DATABASE" on page 124

"GET DATABASE CONFIGURATION" on page 225

"MIGRATE DATABASE" on page 370

"ROLLFORWARD DATABASE" on page 451.

## REWIND TAPE

DB2 for Windows NT supports backup and restore to streaming tape devices. Use this command for tape rewinding.

This command is available on Windows NT only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──REWIND TAPE──┬──────────────┬──────────────────────────────────────►◄
                 └─ON──device───┘
```

### Command Parameters

**ON device**
Specifies a valid tape device name. The default value is `\\.\TAPE0`.

### See Also

“INITIALIZE TAPE” on page 291

“SET TAPE POSITION” on page 474.

## ROLLFORWARD DATABASE

Recovers a database by applying transactions recorded in the database log files. Invoked after a database or a table space backup has been restored, or if any table spaces have been taken offline by the database due to a media error. The database must be recoverable (that is, either *logretain*, *userexit*, or both of these database configuration parameters must be set on) before the database can be recovered with roll-forward recovery.

### Scope

In a multi-node environment, this command can only be issued from the catalog node. A database or table space rollforward command specifying a point-in-time affects all nodes that are listed in the `db2nodes.cfg` file. A database or table space rollforward command specifying end of logs affects the nodes that are specified. If no nodes are specified, it affects all nodes that are listed in the `db2nodes.cfg` file; if no roll forward is needed on a particular node, that node is ignored.

### Authorization

One of the following:

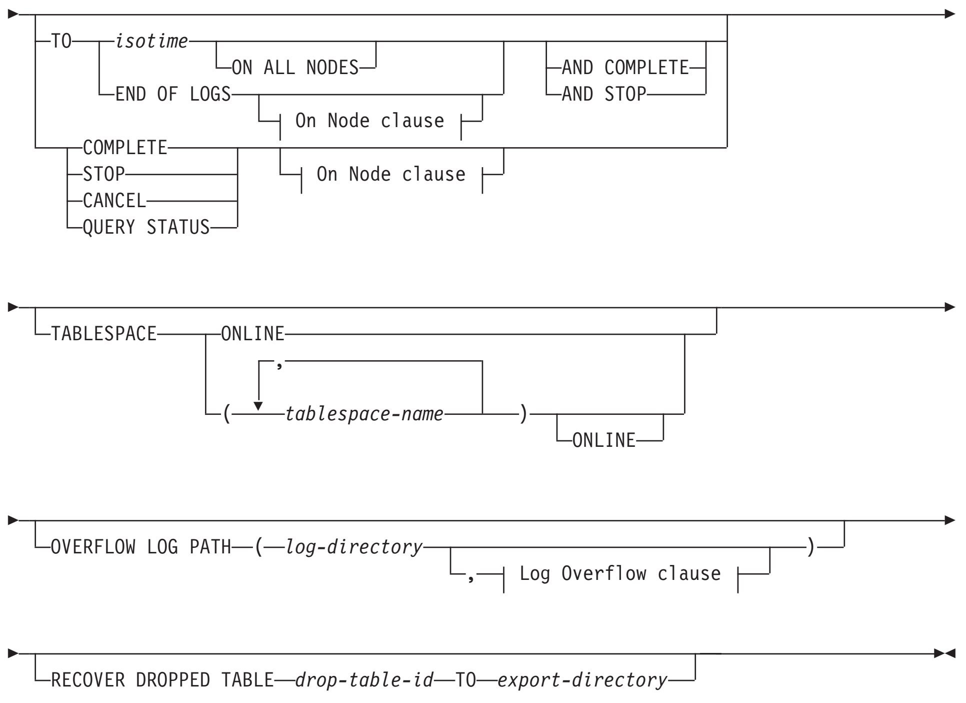
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

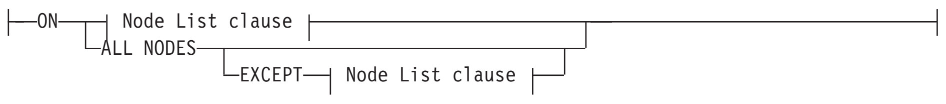
None. This command establishes a database connection.

### Command Syntax

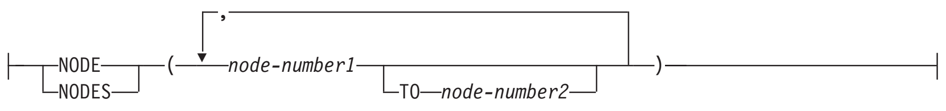
```
►►──ROLLFORWARD──┬─DATABASE─┬──database-alias──────────────────────────────────►
                 └─DB───────┘

►─┬───────────────────────────────────────┬────────────────────────────────────►
  └─USER──username──┬──────────────────┬──┘
                    └─USING──password──┘
```



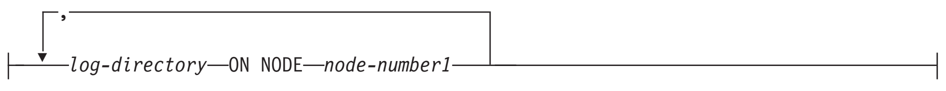
**On Node clause:**



**Node List clause:**



**Log Overflow clause:**



## Command Parameters

**DATABASE database-alias**
The alias of the database to roll forward.

**USER username**
Identifies the user name under which the database is to be rolled forward.

**USING password**
The password used to authenticate the user name. If the password is omitted, the user is prompted to enter it.

**TO**

**isotime**
The point in time to which all committed transactions are to be rolled forward (including the transaction committed precisely at that time, as well as all transactions committed previously).

This value is specified as a time stamp, a 7-part character string that identifies a combined date and time. The format is *yyyy-mm-dd-hh.mm.ss.nnnnnn* (year, month, day, hour, minutes, seconds, microseconds), expressed in Coordinated Universal Time (CUT).

The environment variable **TZ** indicates the difference between CUT and local time. For example, a **TZ** value of `EST5EDT` indicates that the local time zone is EST; that there is a 5-hour difference between this time zone and CUT; and that daylight savings time is observed. This observance reduces the difference from 5 to 4 hours when daylight savings time is in effect, and CUT = current time + 4.

In a partitioned database environment, if ON ALL NODES is specified, roll forward recovery is performed on all nodes.

**END OF LOGS**
Specifies that all committed transactions from all online archive log files listed in the database configuration parameter *logpath* are to be applied.

**ALL NODES**
Specifies that transactions are to be rolled forward on all nodes specified in the `db2nodes.cfg` file. This is the default if a node clause is not specified.

**EXCEPT**
Specifies that transactions are to be rolled forward on all nodes specified in the `db2nodes.cfg` file, except those specified in the node list.

**ON NODE / ON NODES**
Roll forward the database on a set of nodes.

**node-number1**
Specifies a node number in the node list.

**node-number2**
Specifies the second node number, so that all nodes from *node-number1* up to and including *node-number2* are included in the node list.

**COMPLETE / STOP**
Stops the rolling forward of log records, and completes the roll-forward recovery process by rolling back any incomplete transactions and turning off the roll-forward pending state of the database. This allows access to the database or table spaces that are being rolled forward. These keywords are equivalent; specify one or the other, but not both. The keyword **AND** permits specification of multiple operations at once; for example, `db2 rollforward db sample to end of logs and complete`.

**Note:** When rolling table spaces forward to a point-in-time, the table spaces are placed in backup pending state.

**CANCEL**
Cancels the roll-forward recovery process. This leaves the database or table space(s) on all nodes on which forward recovery has been started in the restore-pending state. If the database roll-forward is not in progress (that is, the database is in rollforward-pending state), this option will change the database to restore-pending state. If a table space roll-forward is not in progress (that is, the table spaces are in rollforward-pending state), a table space list must be specified. All table spaces in the list will be changed to restore-pending state. If a table space roll-forward is in progress (that is, at least one table space is in rollforward-in-progress state), all table spaces in rollforward-in-progress state will be changed to restore-pending state. If a table space list is specified, it must include all table spaces in rollforward-in-progress state. If rolling forward to a point in time, any table space passed in will be ignored, and all table spaces that are rollforward-in-progress state will be put in restore pending state. If rolling forward to the end of logs with a table space list, only those table spaces will be put in restore-pending state.

**Note:** Use this option with caution.

**QUERY STATUS**

Lists the log files that the database manager has rolled forward, the next archive file required, and the time stamp (in CUT) of the last committed transaction since roll-forward processing began. In a multi-node environment, this status information is returned for each node. The information returned contains the following fields:

**Node number**

**Rollforward status**

Status may be database or table space rollforward pending, database or table space rollforward in progress, database or table space rollforward processing STOP, or no rollforward pending.

**Next log file to be read**

A string containing the name of the next required log file. In a multi-node environment, use this information if the rollforward utility fails with a return code indicating that a log file is missing or a log information mismatch has occurred.

**Log files processed**

A string containing the names of the processed log files that are no longer needed for recovery, and that can be removed from the directory.

**Last committed transaction**

A string containing a time stamp in ISO format (*yyyy-mm-dd-hh.mm.ss*). This time stamp marks the last transaction committed after the completion of roll-forward recovery. The time stamp applies to the database. For table space roll-forward, it is the time stamp of the last transaction committed to the database.

**Note:** QUERY STATUS is the default if the TO, STOP, COMPLETE, and CANCEL clauses are omitted.

If TO, STOP, or COMPLETE are specified, this information will be displayed if the command ran successfully.

**TABLESPACE**

This keyword is specified for table space level roll-forward.

**tablespace-name**

Mandatory for table space level roll-forward to a point in time. Also allows a subset of table spaces to be specified for a roll-forward to the end of logs. In a multi-node environment, each table space in the list does not have to exist at each node that is rolling forward. If it *does* exist at the node, it must be in the correct state.

**ONLINE**
This keyword is specified to allow the table space level roll-forward recovery to be done online. This means that other agents are allowed to connect while roll-forward recovery is in progress.

**OVERFLOW LOG PATH log-directory**
Specifies an alternate log path to be searched for archived logs during recovery. In a multi-node environment, this is the default overflow log path for all nodes.

In a single-node environment, a relative overflow log path can be specified, but in a multi-node environment, the path must be fully qualified.

**log-directory ON NODE**
In a multi-node environment, allows a different log path to override the default overflow log path for a specific node.

**RECOVER DROPPED TABLE drop-table-id**
Recovers a dropped table during the rollforward operation. The table ID can be obtained using "LIST HISTORY" on page 311.

**TO export-directory**
Specifies a directory to which files containing the table data are to be written. The directory must be accessible to all nodes.

## Examples

**Example 1**

The ROLLFORWARD command permits specification of multiple operations at once, each being separated with the keyword and. For example, to roll forward to the end of logs, and complete, the separate commands:

```
db2 rollforward db sample to end of logs
db2 rollforward db sample complete
```

can be combined as follows:

```
db2 rollforward db sample to end of logs and complete
```

**Example 2**

Roll forward to the end of logs (two table spaces have been restored):

```
db2 rollforward db sample to end of logs
```

**Note:** Neither AND STOP or AND COMPLETE is needed for table space roll forward to the end of logs. Table space names are not required. If not specified, all table spaces requiring roll forward recovery will be included. If only a subset of these table spaces are to be rolled forward, their names must be specified.

**Example 3**

After three table spaces have been restored, roll forward one to the end of logs, and the other two to a point in time, both to be done online:

```
db2 rollforward db sample to end of logs tablespace(TBS1) online

db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   tablespace(TBS2, TBS3) online
```

**Example 4**

After restoring the database, roll forward to a point in time, using OVERFLOW LOG PATH to specify the directory where the user exit saves archived logs:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop
   overflow log path (/logs)
```

**Example 5 (MPP)**

There are three nodes: 0, 1, and 2. Table space TBS1 is defined on all nodes, and table space TBS2 is defined on nodes 0 and 2. After restoring the database on node 1, and TBS1 on nodes 0 and 2, roll forward the database on node 1:

```
db2 rollforward db sample to end of logs and stop
```

This returns warning SQL1271 (″Database is recovered but one or more table spaces are off-line on node(s) 0 and 2.″).

```
db2 rollforward db sample to end of logs
```

This rolls forward TBS1 on nodes 0 and 2. The clause TABLESPACE(TBS1) is optional in this case.

**Example 6 (MPP)**

After restoring table space TBS1 on nodes 0 and 2 only, roll forward TBS1 on nodes 0 and 2:

```
db2 rollforward db sample to end of logs
```

Node 1 is ignored.

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

This fails because TBS1 is not ready for roll forward on node 1. Reports SQL4906N.

```
db2 rollforward db sample to end of logs on nodes (0, 2) tablespace(TBS1)
```

This completes successfully.

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop tablespace(TBS1)
```

This fails because TBS1 is not ready for roll forward on node 1; all pieces must be rolled forward together.

**Note:** With table space roll forward to a point in time, the node clause is not accepted.

After restoring TBS1 on node 1:

```
db2 rollforward db sample to 1998-04-03-14.21.56.245378 and stop tablespace(TBS1)
```

This completes successfully.

**Example 7 (MPP)**

After restoring a table space on all nodes, roll forward to PIT2, but do not specify AND STOP. The ROLLFORWARD command is still in progress. Cancel and roll forward to PIT1:

```
   db2 rollforward db sample to pit2 tablespace(TBS1)
   db2 rollforward db sample cancel tablespace(TBS1)

** restore TBS1 on all nodes **

   db2 rollforward db sample to pit1 tablespace(TBS1)
   db2 rollforward db sample stop tablespace(TBS1)
```

**Example 8 (MPP)**

Roll forward a table space that resides on eight nodes (3 to 10) listed in the db2nodes.cfg file:

```
db2 rollforward database dwtest to end of logs tablespace (tssprodt)
```

This operation to the end of logs (not point in time) completes successfully. The nodes on which the table space resides do not have to be specified. The utility defaults to the db2nodes.cfg file.

**Example 9 (MPP)**

Roll forward six small table spaces that reside on a single node nodegroup (on node 6):

```
db2 rollforward database dwtest to end of logs on node (6)
   tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

This operation to the end of logs (not point in time) completes successfully.

## Usage Notes

The database manager uses the information stored in the archived and the active log files to reconstruct the transactions performed on the database since its last backup.

If the database is in roll-forward pending state when ROLLFORWARD DATABASE is invoked, the database will be rolled forward. Table spaces are returned to normal state after a successful database roll-forward, unless an abnormal state causes one or more table spaces to go offline.

If the database is not in roll-forward pending state and no point in time is specified, any table spaces that are in rollforward-in-progress state will be rolled forward to the end of logs. If no table spaces are in rollforward-in-progress state, any table spaces that are in rollforward pending state will be rolled forward to the end of logs.

The roll-forward operation can be performed on a subset of table spaces by specifying table space names.

If rolling forward table spaces to a point in time, a subset of table spaces must be specified. Only those table spaces specified will be rolled forward. Each table space must be in roll-forward pending state or, if continuing a table space roll-forward that is already in progress, in rollforward-in-progress state.

If enabling an existing database for roll-forward recovery, change the number of primary log files to the sum of the number of primary log files and secondary log files +1. More information will be logged for LONG VARCHAR fields and LOB data in a database enabled for roll-forward recovery.

Rolling databases forward may require a load recovery using tape devices. If prompted for another tape, the user can respond with one of the following:

**c** Continue. Continue using the device that generated the warning message (for example, when a new tape has been mounted)

**d** Device terminate. Stop using the device that generated the warning message (for example, when there are no more tapes)

**t** Terminate. Terminate all devices.

Rolling databases forward may involve prerequisites and restrictions that are beyond the scope of this manual. For more detailed information, see the *Administration Guide*.

## See Also

"LIST HISTORY" on page 311

"LOAD" on page 338

"RESTORE DATABASE" on page 441.

## RUNSTATS

Updates statistics about the physical characteristics of a table and the associated indexes. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

This utility should be called when a table has had many updates, or after reorganizing a table.

### Scope

This command can be issued from any node in the `db2nodes.cfg` file. It can be used to update the catalogs on the catalog node.

The command collects statistics for a table on the node from which it is invoked. If the table does not exist on that node, the first node in the nodegroup is selected.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the table.

### Required Connection

Database

### Command Syntax

```
►►──RUNSTATS ON TABLE──table-name────────────────────────────────────────►

►──┬─┬──────────────────────────────────────────────────────┬─┬────────►
   │ └─WITH DISTRIBUTION─┬───────────────────────────────────┬┘ │
   │                     └─AND──┬──────────┬─┬─INDEXES ALL──────────┬┘  │
   │                            └─DETAILED─┘ └─INDEX──index-name─┘    │
   └─┬─────────────────────────────────────────────────────┬─────┘
     └─┬─AND─┬──┬──────────┬──┬─INDEXES ALL─────────┬─┘
       └─FOR─┘  └─DETAILED─┘  └─INDEX──index-name─┘
```

```
►─┬──────────────────────────────┬──────────────────────────────────────────────►◄
  │                ┌─CHANGE────┐ │
  └─SHRLEVEL───────┴─REFERENCE─┴─┘
```

## Command Parameters

**TABLE table-name**
The table on which to update statistics. The fully qualified name or alias in the form: *schema.table-name* must be used. The *schema* is the user name under which the table was created. If no options are specified, only table statistics will be updated. Other users will have access to the table while the statistics are being gathered.

For row types, *table-name* must be the name of the hierarchy's root table.

**WITH DISTRIBUTION**
Specifies that distribution statistics are requested. The number of most frequent values collected is defined by the *num_freqvalues* database configuration parameter. The number of quantiles collected is defined by the *num_quantiles* database configuration parameter. For information about nonuniform distribution statistics, see the *Administration Guide*.

**AND INDEXES ALL**
Update statistics on both the table and its indexes.

**AND INDEX index-name**
Update statistics on both the table and the specified index, where *index-name* is a fully qualified name in the form: *schema.index-name*.

**FOR INDEXES ALL**
Update statistics on the indexes only. If statistics on the table have never been generated, the database manager calculates statistics on the table as well as on the indexes.

**FOR INDEX index-name**
Update statistics on the specified index only. If table statistics have never been generated, the database manager calculates statistics on the table as well as on the index. The index-name is a fully qualified name in the form: *schema.index-name*.

**DETAILED**
Calculate extended index statistics.

**SHRLEVEL**

**CHANGE**
Specifies that other users can read from and write to the table while statistics are calculated.

**REFERENCE**
Specifies that other users can have read-only access to the table while statistics are calculated.

## Examples

Collect statistics on table only, without distribution statistics:

```
db2 runstats on table smith.table1
```

Collect statistics on table only, with distribution statistics:

```
db2 runstats on table smith.table1 with distribution
```

Collect basic statistics on indexes only:

```
db2 runstats on table smith.table1 for indexes all
```

Collect statistics on table and all indexes (basic level):

```
db2 runstats on table smith.table1 and indexes all
```

Collect statistics on table, with distribution statistics and index statistics:

```
db2 runstats on table smith.table1 with distribution and indexes all
```

Collect all possible statistics (distribution and extended index):

```
db2 runstats on table smith.table1 with distribution and detailed index
```

Collect distribution statistics on index INDEX1 only:

```
db2 runstats on table smith.table1 with distribution for index smith.index1
```

## Usage Notes

This utility does not support the use of nicknames.

Use RUNSTATS to update statistics:

- On tables that have been modified many times (for example, if a large number of updates have been made, or if a significant amount of data has been inserted or deleted)
- On tables that have been reorganized
- When a new index has been created.

After statistics have been updated, new access paths to the table can be created by rebinding the packages using “BIND” on page 131.

If index statistics are requested, and statistics have never been run on the table containing the index, statistics on both the table and indexes are calculated.

After issuing this command, a COMMIT should be issued to release the locks.

To allow new access plans to be generated, the packages that reference the target table must be rebound after issuing this command.

Statistics are collected based on the table data that is located on the database partition where the command executes. Global table statistics for an entire partitioned table are derived by multiplying the values obtained at a database partition by the number of database partitions in the nodegroup over which the table is partitioned. The global statistics are stored in the catalog tables.

The database partition from which the command is issued does not have to contain a partition for the table:

- If the command is issued from a database partition that contains a partition for the table, the utility executes at this database partition.
- If the command is issued from a database partition that does not contain a table partition, the request is sent to the first database partition in the nodegroup that holds a partition for the table. The utility then executes at this database partition.

If inconsistencies are found when running a portion of this command (resulting from activity on the table since the command was last issued), a warning message is returned. For example, if table distribution statistics were gathered on the first issue, and only index statistics are gathered on the second issue, then if inconsistencies are detected as a result of activity on the table, the table distribution statistics are dropped. At this point, it is recommended to issue the command again to refresh the table distribution statistics.

## See Also

"GET DATABASE CONFIGURATION" on page 225

"REORGANIZE TABLE" on page 419

"REORGCHK" on page 422.

## SET CLIENT

Specifies connection settings for the back-end process.

### Authorization

None

### Required Connection

None

### Command Syntax

```
        (1)
>>──────────SET CLIENT──┬───────────────────┬──┬──────────────────────────────┬──────────►
                        └─CONNECT──┬─1─┬────┘  └─DISCONNECT──┬─EXPLICIT────┬──┘
                                   └─2─┘                     ├─CONDITIONAL─┤
                                                             └─AUTOMATIC───┘

►──┬───────────────────────────────────┬──┬──────────────────────┬───────────────────────►
   └─MAX_NETBIOS_CONNECTIONS──value────┘  └─SQLRULES──┬─DB2─┬────┘
                                                      └─STD─┘

►──┬────────────────────────────────┬──┬──────────────────────────────────────┬──────────►
   └─SYNCPOINT──┬─ONEPHASE─┬────────┘  └─CONNECT_NODE──┬─node-number──┬───────┘
                ├─TWOPHASE─┤                           └─CATALOG_NODE─┘
                └─NONE─────┘

►──┬──────────────────────────────┬──────────────────────────────────────────────────────►◄
   └─ATTACH_NODE──node-number─────┘
```

**Notes:**

1. Default is setting unchanged if option is not specified.

### Command Parameters

**CONNECT**

| | |
|---|---|
| **1** | Specifies that a CONNECT statement is to be processed as a type 1 CONNECT. |
| **2** | Specifies that a CONNECT statement is to be processed as a type 2 CONNECT. |

**DISCONNECT**

**EXPLICIT**
Specifies that only database connections that have been explicitly marked for release by the RELEASE statement are to be disconnected at commit.

**CONDITIONAL**
Specifies that the database connections that have been marked RELEASE or have no open WITH HOLD cursors are to be disconnected at commit.

**AUTOMATIC**
Specifies that all database connections are to be disconnected at commit.

**MAX_NETBIOS_CONNECTIONS value**
Specifies the maximum number of concurrent connections that can be made in an application using a NetBIOS adapter. Maximum value is 254. This parameter must be set before the first NetBIOS connection is made. Changes subsequent to the first connection are ignored.

**SQLRULES**

**DB2** Specifies that a type 2 CONNECT is to be processed according to the DB2 rules.

**STD** Specifies that a type 2 CONNECT is to be processed according to the Standard (STD) rules based on ISO/ANS SQL92.

**SYNCPOINT**
Specifies how commits or rollbacks are to be coordinated among multiple database connections.

**ONEPHASE**
Specifies that no Transaction Manager (TM) is to be used to perform a two-phase commit. A one-phase commit is to be used to commit the work done by each database in multiple database transactions.

**TWOPHASE**
Specifies that the TM is required to coordinate two-phase commits among those databases that support this protocol.

**NONE**
Specifies that no TM is to be used to perform a two-phase commit, and does not enforce single updater, multiple reader. A COMMIT is sent to each participating database. The application is responsible for recovery if any of the commits fail.

**CONNECT_NODE (MPP only)**

**node-number**
Specifies the node to which a connect is to be made. A value between zero and 999, inclusive. Overrides the value of the environment variable **DB2NODE**.

**CATALOG_NODE**
Specifying this value permits the client to connect to the catalog node of the database without knowing the identity of that node in advance.

**ATTACH_NODE (MPP only) node-number**
Specifies the node to which an attach is to be made. A value between zero and 999, inclusive. Overrides the value of the environment variable **DB2NODE**.

For example, if nodes 1, 2, and 3 are defined, the client only needs to be able to access one of these nodes. If only node 1 containing databases has been cataloged, and this parameter is set to 3, then the next attach attempt will result in an attachment at node 3, after an initial attachment at node 1.

## Examples

To set specific values:

```
db2 set client connect 2 disconnect automatic sqlrules std
   syncpoint twophase
```

To change SQLRULES back to DB2, but keep the other settings:

```
db2 set client sqlrules db2
```

**Note:** The connection settings revert to default values after “TERMINATE” on page 484 is issued.

## Usage Notes

SET CLIENT cannot be issued if one or more connections are active.

If SET CLIENT is successful, the connections in the subsequent units of work will use the connection settings specified. If SET CLIENT is unsuccessful, the connection settings of the back-end process are unchanged.

For more information about distributed unit of work (DUOW), see the *Administration Guide*.

### See Also

"QUERY CLIENT" on page 397.

## SET RUNTIME DEGREE

Sets the maximum run time degree of intra-partition parallelism for SQL statements for specified active applications.

### Scope

This command affects all nodes that are listed in the `$HOME/sqllib/db2nodes.cfg` file.

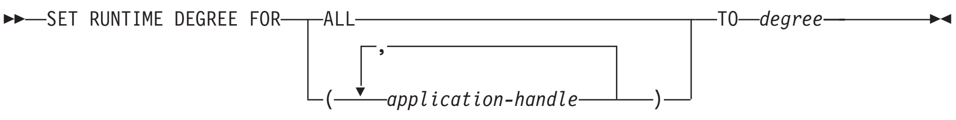
### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

Instance. To change the maximum run time degree of intra-partition parallelism on a remote server, it is first necessary to attach to that server. If no attachment exists, the SET RUNTIME DEGREE command fails.

### Command Syntax



### Command Parameters

**FOR**

**ALL** The specified degree will apply to all applications.

**application-handle**
Specifies the agent to which the new degree applies. List the values using “LIST APPLICATIONS” on page 295.

**TO degree**
The maximum run time degree of intra-partition parallelism.

### Examples

The following example sets the maximum run time degree of parallelism for two users, with *application-handle* values of 41408 and 55458, to 4:

```
db2 SET RUNTIME DEGREE FOR ( 41408, 55458 ) TO 4
```

## Usage Notes

This command provides a mechanism to modify the maximum degree of parallelism for active applications. It can be used to override the value that was determined at SQL statement compilation time.

The run time degree of intra-partition parallelism specifies the maximum number of parallel operations that will be used when the statement is executed. The degree of intra-partition parallelism for an SQL statement can be specified at statement compilation time using the CURRENT DEGREE special register or the **degree** bind option. The maximum run time degree of intra-partition parallelism for an active application can be specified using the SET RUNTIME DEGREE command. The *max_querydegree* database manager configuration parameter specifies the maximum run time degree for any SQL statement executing on this instance of the database manager.

The actual run time degree will be the lowest of:
- the *max_querydegree* configuration parameter
- the application run time degree
- the SQL statement compilation degree.

## SET TABLESPACE CONTAINERS

A *redirected restore* is a restore in which the set of table space containers for the restored database is different from the set of containers for the original database at the time the backup was done. This command permits the addition, change, or removal of table space containers for a database that is to be restored. If, for example, one or more containers become inaccessible for any reason, the restore fails if it is not redirected to different containers.

**Note:** A redirected restore is not allowed when a user exit program is used to perform the restore.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

Database

### Command Syntax

```
►►──SET TABLESPACE CONTAINERS FOR──tablespace-id────────────────────────────►

►──┬─────────────────────────────────────────────────┬──USING───────────────►
   └──┬─REPLAY─┬──ROLLFORWARD CONTAINER OPERATIONS───┘
      └─IGNORE─┘

        ┌─,────────────────────────┐
        ▼                          │
►──┬─(────PATH──"container-string"─┴──)─────────────────────────┬──────────►◄
   │    ┌─,─────────────────────────────────────────────┐       │
   │    ▼                                               │       │
   └─(──────┬─FILE───┬──"container-string"──number-of-pages─┴──)─┘
            └─DEVICE─┘
```

### Command Parameters

**FOR tablespace-id**
: An integer that uniquely represents a table space used by the database being restored.

**REPLAY ROLLFORWARD CONTAINER OPERATIONS**
: Specifies that any ALTER TABLESPACE operation issued against this

table space since the database was backed up is to be redone during a subsequent roll forward of the database.

**IGNORE ROLLFORWARD CONTAINER OPERATIONS**
Specifies that ALTER TABLESPACE operations in the log are to be ignored when performing a roll forward.

**USING PATH ″container-string″**
For an SMS table space, identifies one or more containers that will belong to the table space and into which the table space data will be stored. It is an absolute or relative directory name. If the directory name is not absolute, it is relative to the database directory. The string cannot exceed 240 bytes in length.

**USING FILE/DEVICE ″container-string″ number-of-pages**
For a DMS table space, identifies one or more containers that will belong to the table space and into which the table space data will be stored. The container type (either FILE or DEVICE) and its size (in 4KB pages) are specified. A mixture of file and device containers can be specified. The string cannot exceed 254 bytes in length.

For a file container, the string must be an absolute or relative file name. If the file name is not absolute, it is relative to the database directory.

For a device container, the string must be a device name. The device must already exist. Device containers are not supported on OS/2.

## Examples

See the example in "RESTORE DATABASE" on page 441.

## Usage Notes

This command is used in conjunction with "RESTORE DATABASE" on page 441.

A backup of a database, or one or more table spaces, keeps a record of all the table space containers in use by the table spaces being backed up. During a restore, all containers listed in the backup are checked to see if they currently exist and are accessible. If one or more of the containers is inaccessible for any reason, the restore will fail. In order to allow a restore in such a case, the redirecting of table space containers is supported during the restore. This support includes adding, changing, or removing of table space containers. It is this command that allows the user to add, change or remove those containers. For more information, see the *Administration Guide*.

## See Also

“BACKUP DATABASE” on page 124

“RESTORE DATABASE” on page 441

“ROLLFORWARD DATABASE” on page 451.

## SET TAPE POSITION

DB2 for Windows NT supports backup and restore to streaming tape devices. Use this command for tape positioning.

This command is available on Windows NT only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──SET TAPE POSITION──┬──────────────┬──TO──position──────────────────────────►◄
                       └─ON──device───┘
```

### Command Parameters

**ON device**
: Specifies a valid tape device name. The default value is `\\.\TAPE0`.

**TO position**
: Specifies the mark at which the tape is to be positioned. DB2 for Windows NT writes a tape mark after every backup. A value of 1 specifies the first position, 2 specifies the second position, and so on. If the tape is positioned at tape mark 1, archive 2 is positioned to be restored.

### See Also

"INITIALIZE TAPE" on page 291

"REWIND TAPE" on page 450.

## START DATABASE MANAGER

Starts the current database manager instance background processes on a single node or on all the nodes defined in a multi-node environment.

This command is not valid on a client.

### Scope

In a multi-node environment, this command affects all nodes that are listed in the `$HOME/sqllib/db2nodes.cfg` file, unless the *nodenum* parameter is used.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

**Note:** On OS/2, no authorization is required if the *ss_logon* database manager configuration parameter is set to `NO`.

### Required Connection

None

### Command Syntax

```
►►──┬─START──────┬─DATABASE MANAGER─┬─┬──┬──────────────────┬──────────────────────────►
    │            ├─DB MANAGER───────┤ │  └─PROFILE──profile─┘
    │            └─DBM──────────────┘ │
    └─db2start────────────────────────┘

►──┬─────────────────────────────────────┬──────────────────────────────────────────►◄
   └─NODENUM──nodenum──┤ start options ├─┘
```

**start options:**

```
├──┬─────────────────────────────────────┬────────────────────────────────────────┤
   ├─ADDNODE──┤ addnode options ├────────┤
   ├─STANDALONE──────────────────────────┤
   └─RESTART──┤ restart options ├────────┘
```

**addnode options:**

```
├──HOSTNAME──hostname──PORT──logical-port──┬───────────────────────────┬──────────────►
                                           └─COMPUTER──computer-name──┘

►──┬──────────────────┬──┬──────────────────────┬──┬──────────────────┬──────────────►
   └─USER──username──┘  └─PASSWORD──password──┘  └─NETNAME──netname──┘

►──┬─────────────────────────┬────────────────────────────────────────────────────────┤
   ├─LIKE NODE──node-number──┤
   └─WITHOUT TABLESPACES─────┘
```

**restart options:**

```
├──┬──────────────────────┬──┬───────────────────────┬──┬──────────────────┬──────────┤
   └─HOSTNAME──hostname──┘  └─PORT──logical-port──┘  └─NETNAME──netname──┘
```

## Command Parameters

**Note:** All of the following parameters are valid in an MPP environment only.

**PROFILE profile**
Specifies the name of the profile file to be executed at each node to define the DB2 environment. This file is executed before the nodes are started. The profile file must reside in the `sqllib` directory of the instance owner.

**Note:** The environment variables in the profile file are not necessarily all defined in the user session.

**NODENUM nodenum**
Specifies the node to be started. If no other options are specified, a normal startup is done at this node.

Valid values are from 0 to 999 inclusive. If ADDNODE is not specified, the value must already exist in the `db2nodes.cfg` file of the instance owner. If no node number is specified, all nodes defined in the node configuration file are started.

**ADDNODE**
Specifies that the new node is added to the `db2nodes.cfg` file of the instance owner with the *hostname* and *logical-port* values.

Ensure that the combination of *hostname* and *logical-port* is unique.

The add node utility is executed internally to create all existing databases on the node being added. After a node is added, the `db2nodes.cfg` file is not updated with the new node until a **db2stop** is issued. The node is not part of the MPP system until the next **db2start** following the **db2stop**.

**Note:** When the database partitions are created on the new node, their configuration parameters are set to the default.

**HOSTNAME hostname**
With ADDNODE, specifies the host name to be added to the `db2nodes.cfg` file.

**PORT logical-port**
With ADDNODE, specifies the logical port to be added to the `db2nodes.cfg` file. Valid values are from 0 to 999.

**COMPUTER computer-name**
The computer name for the machine on which the new node is created. This parameter is mandatory on Windows NT, but is ignored on other operating systems.

**USER username**
The user name for the account on the new node. This parameter is mandatory on Windows NT, but is ignored on other operating systems.

**PASSWORD password**
The password for the account on the new node. This parameter is mandatory on Windows NT, but is ignored on other operating systems.

**NETNAME netname**
Specifies the *netname* to be added to the `db2nodes.cfg` file. If not specified, this parameter defaults to the value specified for *hostname*.

**LIKE NODE node-number**
Specifies that the containers for the temporary table spaces will be the same as the containers on the specified *node-number* for each database in the instance. The node specified must be a node that is already in the `db2nodes.cfg` file.

**WITHOUT TABLESPACES**
Specifies that containers for the temporary table spaces are not created for any of the databases. The ALTER TABLESPACE statement must be used to add temporary table space containers to each database before the database can be used.

**STANDALONE**
Specifies that the node is to be started in stand-alone mode. FCM does not attempt to establish a connection to any other node. This option is used when adding a node.

**RESTART**
Starts the database manager after a failure. Other nodes are still operating, and this node attempts to connect to the others. If neither the *hostname* nor the *logical-port* parameter is specified, the database manager is restarted using the *hostname* and *logical-port* values specified in `db2nodes.cfg`. If either parameter is specified, the new values are sent to the other nodes when a connection is established. The `db2nodes.cfg` file is updated with this information.

**HOSTNAME hostname**
With RESTART, specifies the host name to be used to override that in the node configuration file.

**PORT logical-port**
With RESTART, specifies the logical port number to be used to override that in the node configuration file. If not specified, this parameter defaults to the *logical-port* value that corresponds to the *nodenum* value in the `db2nodes.cfg` file. Valid values are from 0 to 999.

**NETNAME netname**
Specifies the *netname* to override that specified in the `db2nodes.cfg` file. If not specified, this parameter defaults to the *netname* value that corresponds to the *nodenum* value in the `db2nodes.cfg` file.

## Examples

The following is sample output from **db2start** issued on a three node system with nodes 10, 20, and 30:

```
04-07-1997 10:33:05    10   0   SQL1063N  DB2START processing was successful.
04-07-1997 10:33:07    20   0   SQL1063N  DB2START processing was successful.
04-07-1997 10:33:07    30   0   SQL1063N  DB2START processing was successful.
SQL1063N  DB2START processing was successful.
```

## Usage Notes

It is not necessary to issue this command on a client node. It is provided for compatibility with older clients, but it has no effect on the database manager.

Once started, the database manager instance runs until the user stops it, even if all application programs that were using it have ended.

If the database manager starts successfully, a successful completion message is sent to the standard output device. If an error occurs, processing stops, and an error message is sent to the standard output device. In a multi-node environment, messages are returned on the node that issued the START DATABASE MANAGER command.

If no parameters are specified in a multi-node database environment, the database manager is started on all parallel nodes using the parameters specified in the node configuration file.

If a START DATABASE MANAGER command is in progress, ensure that the applicable nodes have started *before* issuing a request to the database.

The `db2cshrc` file is not supported and cannot be used to define the environment.

On UNIX platforms, the START DATABASE MANAGER command supports the SIGINT and SIGALRM signals. The SIGINT signal is issued if CTRL+C is pressed. The SIGALRM signal is issued if the value specified for the *start_stop_time* database manager configuration parameter is reached. If either signal occurs, all in-progress startups are interrupted and a message (SQL1044N for SIGINT and SQL6037N for SIGALRM) is returned from each interrupted node to the `$HOME/sqllib/log/db2start.` *timestamp*`.log` error log file. Nodes that are already started are not affected. If CTRL+C is pressed on a node that is starting, **db2stop** must be issued on that node before an attempt is made to start it again.

On the Windows NT operating system, neither the **db2start** command nor the **NET START** command returns warnings if any communication subsystem failed to start. The database manager in a Windows NT environment is implemented as an NT service, and does not return an error if the service is started successfully. Be sure to examine the NT Event Log or the `DB2DIAG.LOG` file for any errors that may have occurred during the running of **db2start**.

## See Also

"ADD NODE" on page 119

"STOP DATABASE MANAGER" on page 480.

## STOP DATABASE MANAGER

Stops the current database manager instance. Unless explicitly stopped, the database manager continues to be active. This command does not stop the database manager instance if any applications are connected to databases. If there are no database connections, but there are instance attachments, it forces the instance attachments and stops the database manager. This command also deactivates any outstanding database activations before stopping the database manager.

On an MPP system, this command stops the current database manager instance on a node or on all nodes. When it stops the database manager on all nodes, it uses the node configuration file `db2nodes.cfg` to obtain information about each node.

This command can also be used to drop a node from the `db2nodes.cfg` file (MPP systems only).

This command is not valid on a client.

### Scope

By default, and in a multi-node environment, this command affects all nodes that are listed in the `db2nodes.cfg` file.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

**Note:** On OS/2, no authorization is required if the *ss_logon* database manager configuration parameter is set to `NO`.

### Required Connection

None

### Command Syntax

```
►►──┬─STOP──┬─DATABASE MANAGER─┬─┬──┬──────────────────┬────────────►
    │       ├─DB MANAGER───────┤ │  └─PROFILE──profile─┘
    │       └─DBM──────────────┘ │
    └─db2stop────────────────────┘
```

```
►─┬──────────────────────────────┬──────────────────────────────────────────────►◄
  ├─NODENUM──nodenum─────────────┤
  ├─DROP NODENUM──nodenum────────┤
  └─FORCE──┬───────────────────┬─┘
           └─NODENUM──nodenum──┘
```

## Command Parameters

**PROFILE profile**
: MPP only. Specifies the name of the profile file that was executed at startup to define the DB2 environment for those nodes that were started. If a profile for "START DATABASE MANAGER" on page 475 was specified, the same profile must be specified here. The profile file must reside in the `sqllib` directory of the instance owner.

**NODENUM nodenum**
: MPP only. Specifies the node to be stopped.

Valid values are from 0 to 999 inclusive, and must be in the `db2nodes.cfg` file. If no node number is specified, all nodes defined in the node configuration file are stopped.

**DROP NODENUM nodenum**
: MPP only. Specifies the node to be dropped from the `db2nodes.cfg` file.

Before using this parameter, run "DROP NODE VERIFY" on page 204 to ensure that there is no user data on this node.

When this option is specified, all nodes in the `db2nodes.cfg` file are stopped.

**FORCE**
: Specifies to use FORCE APPLICATION ALL when stopping the database manager at each node.

**NODENUM nodenum**
: MPP only. Specifies the node (database partition server) to be stopped after all applications on that node have been forced to stop. If the FORCE option is used without this parameter, all applications on all nodes are forced before all the nodes are stopped.

## Examples

The following is sample output from **db2stop** issued on a three node system with nodes 10, 20, and 30:

```
04-07-1997 10:32:53    10   0   SQL1064N  DB2STOP processing was successful.
04-07-1997 10:32:54    20   0   SQL1064N  DB2STOP processing was successful.
04-07-1997 10:32:55    30   0   SQL1064N  DB2STOP processing was successful.
SQL1064N  DB2STOP processing was successful.
```

## Usage Notes

It is not necessary to issue this command on a client node. It is provided for compatibility with older clients, but it has no effect on the database manager.

Once started, the database manager instance runs until the user stops it, even if all application programs that were using it have ended.

If the database manager is stopped, a successful completion message is sent to the standard output device. If an error occurs, processing stops, and an error message is sent to the standard output device.

If the database manager cannot be stopped because application programs are still connected to databases, use "FORCE APPLICATION" on page 215 to disconnect all users first, or reissue the STOP DATABASE MANAGER command with the FORCE option.

The following information currently applies to multi-node environments only:

- If no parameters are specified, the database manager is stopped on each node listed in the node configuration file. The `db2diag.log` file may contain messages to indicate that other nodes are shutting down.
- Any nodes added to the MPP system since the previous STOP DATABASE MANAGER command was issued will be updated in the `db2nodes.cfg` file.
- On UNIX platforms, this command supports the SIGALRM signal, which is issued if the value specified for the *start_stop_time* database manager configuration parameter is reached. If this signal occurs, all in-progress stops are interrupted, and message SQL6037N is returned from each interrupted node to the `$HOME/sqllib/log/db2stop.` *timestamp*`.log` error log file. Nodes that are already stopped are not affected.
- The `db2cshrc` file is not supported and cannot be specified as the value for the PROFILE parameter.

**Attention:** The UNIX **kill** command should *not* be used to terminate the database manager because it will abruptly end database manager processes without controlled termination and cleanup processing.

## See Also

"DEACTIVATE DATABASE" on page 194

"DROP NODE VERIFY" on page 204

"FORCE APPLICATION" on page 215

"START DATABASE MANAGER" on page 475.

## TERMINATE

Explicitly terminates the command line processor's back-end process. For more information about back-end and front-end processes, see "Command Line Processor Design" on page 106.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──TERMINATE──────────────────────────────────────────────────────────►◄
```

### Command Parameters

None

### Usage Notes

If an application is connected to a database, or a process is in the middle of a unit of work, TERMINATE causes the database connection to be lost. An internal commit is then performed.

Although TERMINATE and CONNECT RESET both break the connection to a database, only TERMINATE results in termination of the back-end process.

It is recommended that TERMINATE be issued if **db2start** and **db2stop** were executed while the back-end process was active. This prevents the back-end process from maintaining an attachment to a database manager instance that is no longer available.

Back-end processes in MPP systems must also be terminated when the **DB2NODE** environment variable is updated in the session. This environment variable is used to specify the coordinator node number within an MPP multiple logical node configuration.

## UNCATALOG DATABASE

Deletes a database entry from the system database directory.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*

### Required Connection

None. Directory operations affect the local directory only.

### Command Syntax

```
►►──UNCATALOG──┬─DATABASE─┬──database-alias────────────────────────────────────►◄
               └─DB───────┘
```

### Command Parameters

**DATABASE database-alias**
Specifies the alias of the database to uncatalog.

### Usage Notes

Only entries in the system database directory can be uncataloged. Entries in the local database directory can be deleted using “DROP DATABASE” on page 202.

To recatalog the database, use “CATALOG DATABASE” on page 153. To list the databases that are cataloged on a node, use “LIST DATABASE DIRECTORY” on page 299.

The authentication type of a database, used when communicating with a down-level server, can be changed by first uncataloging the database, and then cataloging it again with a different type.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database. To refresh the directory cache for another application, stop and then restart that application.

## UNCATALOG DCS DATABASE

Deletes an entry from the Database Connection Services (DCS) directory.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None. Directory operations affect the local directory only.

### Command Syntax

```
>>—UNCATALOG DCS——┬DATABASE┬——database-alias——————————————————————————————><
                  └DB——————┘
```

### Command Parameters

**DATABASE database-alias**
Specifies the alias of the DCS database to uncatalog.

### Usage Notes

DCS databases are also cataloged in the system database directory as remote databases that can be uncataloged using "UNCATALOG DATABASE" on page 485.

To recatalog a database in the DCS directory, use "CATALOG DCS DATABASE" on page 157. To list the DCS databases that are cataloged on a node, use "LIST DCS DIRECTORY" on page 306.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in "GET DATABASE MANAGER CONFIGURATION" on page 236), database, node, and DCS directory files are cached in memory. An application's directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP's directory cache, use "TERMINATE" on page 484. To refresh DB2's shared cache, stop (**db2stop**) and then restart (**db2start**) the database. To refresh the directory cache for another application, stop and then restart that application.

## UNCATALOG LDAP DATABASE

Used to deregister the database from LDAP (Lightweight Directory Access Protocol).

This command is available on Windows NT, Windows 98, and Windows 95 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──UNCATALOG LDAP──┬─DATABASE─┬──dbalias──────────────────────────────────►
                    └─DB───────┘

►─┬────────────────────────────────────────┬──────────────────────────────►◄
  └─USER──username──┬────────────────────┬─┘
                    └─PASSWORD──password─┘
```

### Command Parameters

**DATABASE dbalias**
: Specifies the alias of the LDAP database to uncatalog.

**USER username**
: Specifies the user's LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to delete the object from the LDAP directory. If the user's LDAP DN is not specified, the credentials of the current logon user will be used.

**PASSWORD password**
: Account password.

### Usage Notes

When a database is dropped, the database object is removed from LDAP. The database is also automatically deregistered from LDAP when the database server that manages the database is deregistered from LDAP. It may, however, be necessary to manually uncatalog the database from LDAP if:

- The database server does not support LDAP. The administrator must manually uncatalog each database from LDAP after the database is dropped.

- During DROP DATABASE the database object cannot be removed from LDAP (because LDAP cannot be accessed). In this case, the database is still removed from the local machine, but the existing entry in LDAP is not deleted.

## See Also

"CATALOG LDAP DATABASE" on page 165

"CATALOG LDAP NODE" on page 169

"UNCATALOG LDAP NODE" on page 490.

## UNCATALOG LDAP NODE

Uncatalogs a node entry in LDAP (Lightweight Directory Access Protocol).

This command is available on Windows NT, Windows 98, and Windows 95 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──UNCATALOG LDAP──NODE──nodename──┬──────────────────────────────────────────┬──────►◄
                                    └─USER──username──┬──────────────────────┬─┘
                                                      └─PASSWORD──password──┘
```

### Command Parameters

**NODE nodename**
: Specifies the name of the node to uncatalog.

**USER username**
: Specifies the user's LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to delete the object from the LDAP directory. If the user's LDAP DN is not specified, the credentials of the current logon user will be used.

**PASSWORD password**
: Account password.

### Usage Notes

The LDAP node is automatically uncataloged when the DB2 server is deregistered from LDAP.

### See Also

"CATALOG LDAP DATABASE" on page 165

"CATALOG LDAP NODE" on page 169

"UNCATALOG LDAP DATABASE" on page 488.

## UNCATALOG NODE

Deletes an entry from the node directory.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*

### Required Connection

None. Directory operations affect the local directory only.

### Command Syntax

```
►►──UNCATALOG NODE──nodename──────────────────────────────────────────────►◄
```

### Command Parameters

**NODE nodename**
Specifies the node entry being uncataloged.

### Usage Notes

UNCATALOG NODE can be executed on any type of node, but only the local directory is affected, even if there is an attachment to a remote instance, or a different local instance.

**Note:** If directory caching is enabled (see the configuration parameter *dir_cache* in “GET DATABASE MANAGER CONFIGURATION” on page 236), database, node, and DCS directory files are cached in memory. An application’s directory cache is created during its first directory lookup. Since the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications may not be effective until the application has restarted.

To refresh the CLP’s directory cache, use “TERMINATE” on page 484. To refresh DB2’s shared cache, stop (**db2stop**) and then restart (**db2start**) the database. To refresh the directory cache for another application, stop and then restart that application.

## See Also

“CATALOG APPC NODE” on page 145

“CATALOG APPCLU NODE” on page 148

“CATALOG APPN NODE” on page 150

“CATALOG IPX/SPX NODE” on page 162

“CATALOG LOCAL NODE” on page 171

“CATALOG NETBIOS NODE” on page 175

“CATALOG NAMED PIPE NODE” on page 173

“CATALOG TCP/IP NODE” on page 179.

## UNCATALOG ODBC DATA SOURCE

Uncatalogs a user or system ODBC data source.

A *data source*, in ODBC (Open Database Connectivity) terminology, is a user-defined name for a specific database or file system. That name is used to access the database or file system through ODBC APIs. On Windows NT and Windows 95, either user or system data sources can be uncataloged. A user data source is only visible to the user who cataloged it, whereas a system data source is visible to and can be used by all other users.

This command is available on Windows NT, Windows 95, and Windows 3.1 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
                 ┌─USER───┐
►►──UNCATALOG────┼────────┼──ODBC DATA SOURCE──data-source-name──────────────────►◄
                 └─SYSTEM─┘
```

### Command Parameters

**USER** Uncatalog a user data source. This is the default if no keyword is specified.

**SYSTEM**
Uncatalog a system data source.

**ODBC DATA SOURCE data-source-name**
Specifies the name of the data source to be uncataloged. Maximum length is 32 characters.

### See Also

"CATALOG ODBC DATA SOURCE" on page 178

"LIST ODBC DATA SOURCES" on page 325.

## UPDATE ADMIN CONFIGURATION

Modifies individual entries in the database manager configuration file that are relevant to the DB2 Administration Server. The DB2 Administration Server is a special DB2 instance that enables remote administration of DB2 servers. The following database manager configuration parameters can be modified:

- AGENT_STACK_SZ
- AUTHENTICATION
- DIAGLEVEL
- DIAGPATH
- DISCOVER
- DISCOVER_COMM
- FILESERVER
- IPX_SOCKET
- NNAME
- OBJECTNAME
- QUERY_HEAP_SZ
- SYSADM_GROUP
- SYSCTRL_GROUP
- SYSMAINT_GROUP
- TPNAME
- TRUST_ALLCLNTS
- TRUST_CLNTAUTH

**Note:** It is not recommended that the SVCENAME parameter, set by the installation program, be modified by the user. The administration server service name is set to use the DB2 registered TCP/IP port (523).

For more information about these parameters, see "GET DATABASE MANAGER CONFIGURATION" on page 236.

### Scope

This command can be issued from any node listed in the `db2nodes.cfg` file. It affects all nodes that are listed in this file.

### Authorization

*sysadm*

## Required Connection

None or instance. An instance attachment is not required to perform local DBM configuration operations, but is required to perform remote DBM configuration operations. To update the database manager configuration for a remote instance, it is necessary to first attach to that instance.

## Command Syntax

```
                                                     ┌──────────────────────────┐
                                                     ▼                          │
►►──UPDATE ADMIN──┬─CONFIGURATION─┬──USING────config-keyword value──┴──────────────►◄
                  ├─CONFIG────────┤
                  └─CFG───────────┘
```

## Command Parameters

**USING config-keyword value**
Specifies the admin configuration parameter to be updated.

## Usage Notes

To view or print a list of the admin configuration parameters, use "GET ADMIN CONFIGURATION" on page 217.

To reset the admin configuration parameters to the recommended database manager defaults, use "RESET ADMIN CONFIGURATION" on page 430.

For more information about admin configuration parameters, see the *Administration Guide*.

The values of these parameters differ for each type of database node configured (server, client, or server with remote clients). See the *Administration Guide*, or one of the *Quick Beginnings* books for the ranges and the default values that can be set on each node type.

Changes to the database manager configuration file become effective only after they are loaded into memory. This occurs during execution of **db2start**.

If an error occurs, the database manager configuration file does not change.

The database manager configuration file cannot be updated if the checksum is invalid. This may occur if the database manager configuration file is changed without using the appropriate command. If this happens, the database manager must be reinstalled to reset the database manager configuration file.

### See Also

"GET ADMIN CONFIGURATION" on page 217

"RESET ADMIN CONFIGURATION" on page 430.

## UPDATE CLI CONFIGURATION

Updates the contents of a specified section in the `db2cli.ini` file.

The `db2cli.ini` file is used as the DB2 call level interface (CLI) configuration file. It contains various keywords and values that can be used to modify the behavior of the DB2 CLI and the applications using it. The file is divided into sections, each section corresponding to a database alias name. For more information about this file and the CLI/ODBC configuration keywords, see the *CLI Guide and Reference*.

### Authorization

*sysadm*

### Required Connection

None

### Command Syntax

```
►►──UPDATE CLI──┬─CONFIGURATION─┬──FOR SECTION──section-name──┬──────────────┬──────────►
                ├─CONFIG────────┤                             └─AT USERLEVEL─┘
                └─CFG───────────┘


      ┌──────────────┐
      ▼              │
►─USING───keyword value─┴────────────────────────────────────────────────────────────►◄
```

### Command Parameters

**FOR SECTION section-name**
: Name of the section whose keywords are to be updated. If the specified section does not exist, a new section is created.

**AT USERLEVEL**
: Specifies that the CLI configuration parameter is to be updated at the user level. This parameter is only applicable when LDAP support is enabled.

**USING keyword value**
: Specifies the CLI/ODBC parameter to be updated.

### Usage Notes

The section name and the keywords specified on this command are not case sensitive. However, the keyword values *are* case sensitive.

If a keyword value is a string containing single quotation marks or imbedded blanks, the entire string must be delimited by double quotation marks. For example:

```
   db2 update cli cfg for section tstcli1x
      using TableType "'TABLE','VIEW','SYSTEM TABLE'"
```

When the AT USERLEVEL keywords are specified, the CLI configuration parameters for the specified section are updated only for the current user; otherwise, they are updated for all users on the local machine. The CLI configuration at the user level is maintained in the LDAP directory and cached on the local machine. When reading the CLI configuration, DB2 always reads from the cache. The cache is refreshed when:

- The user updates the CLI configuration.
- The user explicitly forces a refresh of the CLI configuration using the REFRESH LDAP command.

## See Also

"GET CLI CONFIGURATION" on page 222

"REFRESH LDAP" on page 413.

## UPDATE COMMAND OPTIONS

Sets one or more command options during an interactive session, or from a batch input file. The settings revert to system defaults (or the system default overrides in **DB2OPTIONS**) when the interactive session or batch input file ends.

### Authorization

None

### Required Connection

None

### Command Syntax

```
                                 ┌─────────────────────────────────────────┐
                                 ▼                                         │
►►──UPDATE COMMAND OPTIONS USING───option-letter──┬─ON──value─┬────────────┴──────────────────►◄
                                                  └─OFF───────┘
```

### Command Parameters

**USING option-letter**
The following option-letters can be set:

| | |
|---|---|
| **a** | Display SQLCA |
| **c** | Auto-commit SQL statements |
| **e** | Display SQLCODE/SQLSTATE |
| **l** | Log commands in a history file |
| **o** | Display to standard output |
| **p** | Display DB2 interactive prompt |
| **r** | Save output report to a file |
| **s** | Stop execution on command error |
| **v** | Echo current command |
| **w** | Show SQL statement warning messages |
| **z** | Redirect all output to a file. |

**ON value**
The `e`, `l`, `r`, and `z` options require a value if they are turned on. For the `e` option, *value* can be `c` to display the SQLCODE, or `s` to display

the SQLSTATE. For the `l`, `r`, and `z` options, *value* represents the name to be used for the history file or the report file. No other options accept a value.

## Usage Notes

These settings override system defaults, settings in **DB2OPTIONS**, and options specified using the command line option flags.

The file input option (`-f`) and the statement termination option (`-t`) cannot be updated using this command.

To view the current option settings, use “LIST COMMAND OPTIONS” on page 297.

For detailed information about these options, see “Command Line Processor Options” on page 97.

## UPDATE DATABASE CONFIGURATION

Modifies individual entries in a specific database configuration file.

A database configuration file resides on every node on which the database has been created.

### Scope

This command only affects the node on which it is executed.

### Authorization

One of the following:
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance. An explicit attachment is not required. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command.

### Command Syntax

```
►►──UPDATE──┬─DATABASE─┬──┬─CONFIGURATION─┬──FOR──database-alias──────────────────►
            └─DB───────┘  ├─CONFIG────────┤
                          └─CFG───────────┘

             ┌─────────────────────┐
►─USING──────▼─config-keyword value─┴──────────────────────────────────────────────►◄
```

### Command Parameters

**FOR database-alias**
: Specifies the alias of the database whose configuration is to be updated.

**USING config-keyword value**
: Specifies the database configuration parameter to be updated. For a brief description of configurable parameters, see "GET DATABASE CONFIGURATION" on page 225.

## Usage Notes

To view or print a list of the database configuration parameters, use “GET DATABASE CONFIGURATION” on page 225.

To reset the database configuration parameters to the recommended database manager defaults, use “RESET DATABASE CONFIGURATION” on page 433.

For more information about DB2 configuration parameters, see the *Administration Guide*.

The values of these parameters differ for each type of database node configured (server, client, or server with remote clients). See the *Administration Guide* for the ranges and the default values that can be set on each node type.

Not all parameters can be updated.

Most changes to the database configuration file become effective only after they are loaded into memory. All applications must disconnect from the database before this can occur.

If an error occurs, the database configuration file does not change.

The database configuration file cannot be updated if the checksum is invalid. This may occur if the database configuration file is changed without using the appropriate command. If this happens, the database must be restored to reset the database configuration file.

## See Also

“GET DATABASE CONFIGURATION” on page 225

“RESET DATABASE CONFIGURATION” on page 433.

## UPDATE DATABASE MANAGER CONFIGURATION

Modifies individual entries in the database manager configuration file.

### Authorization

*sysadm*

### Required Connection

None or instance. An instance attachment is not required to perform local DBM configuration operations, but is required to perform remote DBM configuration operations. To update the database manager configuration for a remote instance, it is necessary to first attach to that instance.

### Command Syntax

```
►►──UPDATE──┬─DATABASE MANAGER─┬──┬─CONFIGURATION─┬──────────────────────────────►
            ├─DB MANAGER───────┤  ├─CONFIG────────┤
            └─DBM──────────────┘  └─CFG───────────┘

              ┌──────────────────────┐
              ▼                      │
►─USING────────config-keyword value──┴──────────────────────────────────────────►◄
```

### Command Parameters

**USING config-keyword value**
: Specifies the database manager configuration parameter to be updated. For a brief description of configurable parameters, see "GET DATABASE MANAGER CONFIGURATION" on page 236.

### Usage Notes

To view or print a list of the database manager configuration parameters, use "GET DATABASE MANAGER CONFIGURATION" on page 236.

To reset the database manager configuration parameters to the recommended database manager defaults, use "RESET DATABASE MANAGER CONFIGURATION" on page 435.

For more information about database manager configuration parameters, see the *Administration Guide*.

The values of these parameters differ for each type of database node configured (server, client, or server with remote clients). See the *Administration Guide* for the ranges and the default values that can be set on each node type.

Not all parameters can be updated.

Most changes to the database manager configuration file become effective only after they are loaded into memory. For a server configuration parameter, this occurs during execution of **db2start**. For a client configuration parameter, this occurs when the application is restarted. If the client is the command line processor, it is necessary to invoke "TERMINATE" on page 484.

If an error occurs, the database manager configuration file does not change.

The database manager configuration file cannot be updated if the checksum is invalid. This may occur if the database manager configuration file is changed without using the appropriate command. If this happens, the database manager must be reinstalled to reset the database manager configuration file.

## See Also

"GET DATABASE MANAGER CONFIGURATION" on page 236

"RESET DATABASE MANAGER CONFIGURATION" on page 435.

## UPDATE LDAP NODE

Updates the protocol information associated with a node entry that represents the DB2 server in LDAP (Lightweight Directory Access Protocol).

This command is available on Windows NT, Windows 98, and Windows 95 only.

### Authorization

None

### Required Connection

None

### Command Syntax

```
►►──UPDATE LDAP──NODE──nodename──┬─────────────────────┬──┬─────────────────────┬──────►
                                 └─HOSTNAME──hostname──┘  └─SVCENAME──svcename──┘

►──┬───────────────┬──┬────────────────────┬──┬─────────────────────────┬───────────►
   └─NNAME──nname──┘  └─NETWORK──net_id────┘  └─PARTNERLU──partner_lu───┘

►──┬─────────────────┬──┬─────────────┬──┬─────────────────────────┬────────────────►
   └─TPNAME──tpname──┘  └─MODE──mode──┘  └─SECURITY──┬─NONE────┬───┘
                                                     ├─SAME────┤
                                                     └─PROGRAM─┘

►──┬─────────────────────────┬──┬───────────────────────────────────┬───────────────►
   └─LANADDRESS──lan_address─┘  └─CHGPWDLU──change_password_lu──────┘

►──┬───────────────────────┬──┬────────────────────┬────────────────────────────────►
   └─IPX_ADDRESS──ipxaddr──┘  └─WITH──"comments"───┘

►──┬────────────────────────────────────────────────┬──────────────────────────────►◄
   └─USER──username──┬─────────────────────────┬────┘
                     └─PASSWORD──password──────┘
```

### Command Parameters

**NODE nodename**
: Specifies the node name when updating a remote DB2 server. The node name is the value specified when registering the DB2 server in LDAP.

**HOSTNAME hostname**
Specifies the TCP/IP host name (or IP address).

**SVCENAME svcename**
Specifies the TCP/IP service name or port number.

**NNAME nname**
Specifies the NetBIOS workstation name.

**NETWORK net_id**
Specifies the APPN network ID.

**PARTNERLU partner_lu**
Specifies the APPN partner LU name for the DB2 server machine.

**TPNAME tpname**
Specifies the APPN transaction program name.

**MODE mode**
Specifies the APPN mode name.

**SECURITY**
Specifies the APPN security level. Valid values are:

**NONE**
Specifies that no security information is to be included in the allocation request sent to the server. This is the default security type for DB2 UDB server.

**SAME** Specifies that a user name is to be included in the allocation request sent to the server, together with an indicator that the user name has been ″already verified″. The server must be configured to accept ″already verified″ security.

**PROGRAM**
Specifies that both a user name and a password are to be included in the allocation request sent to the server. This is the default security type for host database servers such as DB2 for MVS, DB2 for AS/400, or DB2 for VM.

**LANADDRESS lan_address**
Specifies the APPN network adaptor address.

**CHGPWDLU change_password_lu**
Specifies the name of the partner LU that is to be used when changing the password for a host database server.

**IPX_ADDRESS ipxaddr**
Specifies the complete IPX address. The IPX address of a system on which DB2 UDB is installed can be found by invoking the **db2ipxad** command. The IPX address consists of a 12-digit network address, an

8-digit node address, and a 4-digit socket number: <*NetworkAddress*>.<*NodeAddress*>.<*socket*>

**WITH ″comments″**
Describes the DB2 server. Any comment that helps to describe the server registered in the network directory can be entered. Maximum length is 30 characters. A carriage return or a line feed character is not permitted. The comment text must be enclosed by double quotation marks.

**USER username**
Specifies the user’s LDAP distinguished name (DN). The LDAP user DN must have sufficient authority to create and update the object in the LDAP directory. If the user’s LDAP DN is not specified, the credentials of the current logon user will be used.

**PASSWORD password**
Account password.

## See Also

“db2ipxad - Get IPX/SPX Internetwork Address” on page 48

“DEREGISTER” on page 196

“REGISTER” on page 414.

## UPDATE MONITOR SWITCHES

Turns one or more database monitor recording switches on or off. When the database manager starts, the settings of the six switches are determined by the *dft_mon* database manager configuration parameters (see “GET DATABASE MANAGER CONFIGURATION” on page 236).

The database monitor records a base set of information at all times. Users who require more than this basic information can turn on the appropriate switches, but at a cost to system performance. The amount of information available in output from “GET SNAPSHOT” on page 254 reflects which, if any, switches are on.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

### Required Connection

Instance or database:

- If there is neither an attachment to an instance, nor a connection to a database, a default instance attachment is created.
- If there is both an attachment to an instance, and a database connection, the instance attachment is used.

To update the monitor switches at a remote instance (or a different local instance), it is necessary to first attach to that instance.

### Command Syntax

```
                                   ┌─────────────────────────────┐
                                   ▼                             │
►►──UPDATE MONITOR SWITCHES USING────switch-name──┬─ON──┬──┴──────────────────────────►◄
                                                  └─OFF─┘
```

### Command Parameters

**USING switch-name**
The following switch names are available:

**BUFFERPOOL**
Buffer pool activity information

| | |
|---|---|
| **LOCK** | Lock information |
| **SORT** | Sorting information |
| **STATEMENT** | SQL statement information |
| **TABLE** | Table activity information |
| **UOW** | Unit of work information. |

## Usage Notes

Information is collected by the database manager only after a switch is turned on. The switches remain set until **db2stop** is issued, or the application that issued the UPDATE MONITOR SWITCHES command terminates. To clear the information related to a particular switch, set the switch off, then on.

Updating switches in one application does not affect other applications.

To view the switch settings, use “GET MONITOR SWITCHES” on page 252.

## UPDATE RECOVERY HISTORY FILE

Updates the location, device type, or comment in a recovery history file entry.

### Authorization

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

### Required Connection

Database

### Command Syntax

```
►►──UPDATE HISTORY FOR──object-part──WITH──────────────────────────────────►

►──┬─LOCATION──new-location──DEVICE TYPE──new-device-type─┬──────────────►◄
   └─COMMENT──new-comment─────────────────────────────────┘
```

### Command Parameters

**FOR object-part**
: Specifies the identifier for the backup or copy image. It is a time stamp with a sequence number from 001 to 999.

**LOCATION new-location**
: Specifies the new physical location of a backup. The interpretation of this parameter depends on the device type.

**DEVICE TYPE new-device-type**
: Specifies a new device type for storing the backup. Valid device types are:

| | |
|---|---|
| **D** | Disk |
| **K** | Diskette |
| **T** | Tape |
| **A** | ADSM |
| **U** | User exit |
| **O** | Other. |

**COMMENT new-comment**
Specifies a new comment to describe the entry.

## Examples

To update the history file entry for the full database backup taken on April 13, 1997 at 10:00 a.m., enter:

```
db2 update history for 19970413100000001 with
   location /backup/dbbackup.1 device type d
```

## Usage Notes

The recovery history file is used for record keeping purposes only. It is not used during database recovery.

## See Also

“PRUNE HISTORY/LOGFILE” on page 395.

# Chapter 4. Using Command Line SQL Statements

This section provides information about using Structured Query Language (SQL) statements from the command line. These statements can be executed directly from an operating system command prompt, and can be used to define and manipulate information stored in a database table, index, or view in much the same way as if the commands were written into an application program. Information can be added, deleted, or updated, and reports can be generated from the contents of tables.

All SQL statements that can be executed through the command line processor are listed in the CLP column of Table 8 on page 518. The syntax of all the SQL statements, whether executed from the command line or embedded in a source program, is described in the *SQL Reference*. The syntax of many embedded SQL statements and CLP SQL statements is identical. However, host variables, parameter markers, descriptor names, and statement names are applicable only to embedded SQL. The syntax of CLOSE, CONNECT, DECLARE CURSOR, FETCH, OPEN, and SELECT *does* depend on whether these statements are embedded or executed through the CLP. The CLP syntax of these statements is provided below:

**CALL**

```
►►──CALL──┬─SQLJ.INSTALL_JAR──(──jar-url, jar-id──)─┬─────────────────────────────►◄
          ├─SQLJ.REPLACE_JAR──(──jar-url, jar-id──)─┤
          └─SQLJ.REMOVE_JAR──(──jar-id──)───────────┘
```

**CLOSE**

```
►►──CLOSE──cursor-name──────────────────────────────────────────────────────────►◄
```

**CONNECT**

```
►►──CONNECT──┬──────────────────────────────────────────────────────────────┬───►◄
             ├─TO──server-name──┬────────────────┬──┬───────────────────┬────┤
             │                  └─┤ lock-block ├─┘  └─┤ authorization ├─┘    │
             ├─RESET─────────────────────────────────────────────────────────┤
             │                     (1)                                       │
             └─┤ authorization ├─────────────────────────────────────────────┘
```

**authorization:**

```
├──USER──authorization-name─────────────────────────────────────────────►

►──┬────────────────────────────────────────────────────┬──────────────┤
   ├─USING──password─┬─────────────────────────────────┬┤
   │                 └─NEW──password──CONFIRM──password─┘│
   └─CHANGE PASSWORD─────────────────────────────────────┘
```

**lock-block:**

```
   ┌─IN SHARE MODE──────────────────┐
├──┼────────────────────────────────┼─────────────────────────────────┤
   └─IN EXCLUSIVE MODE─┬────────────────┬┘
                       └─ON SINGLE NODE─┘
```

**Notes:**

1. This form is only valid if implicit connect is enabled.

**DECLARE CURSOR**

```
►►──DECLARE──cursor-name──CURSOR──┬───────────┬──FOR──select-statement──────────►◄
                                  └─WITH HOLD─┘
```

**FETCH**

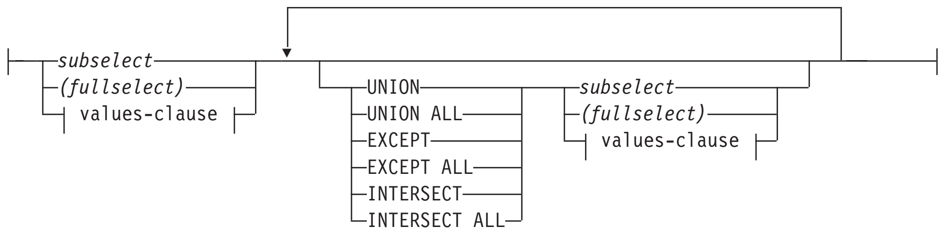
```
►►──FETCH──┬──────┬──cursor-name───────────────────────────────────────►
           └─FROM─┘

►──┬───────────────────────────────────────────────────────┬──────────►◄
   ├─FOR──┬─ALL─┬──┬─ROW──┬────────────────────────────────┤
   │      └─n───┘  └─ROWS─┘                                │
   └─LOB──┬─COLUMN──┬──ALL──INTO──filename──┬─APPEND────┬──┘
          └─COLUMNS─┘                       ├─NEW───────┤
                                            └─OVERWRITE─┘
```

**OPEN**

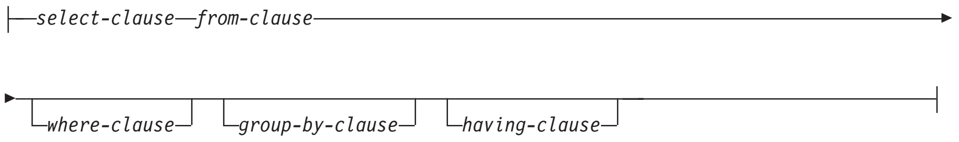
```
►►──OPEN──cursor-name──────────────────────────────────────────────────►◄
```

## SELECT

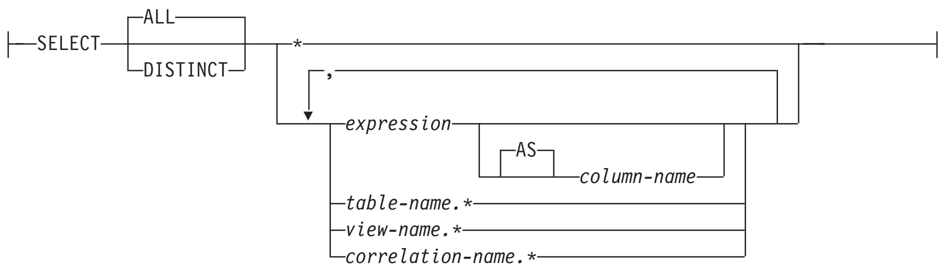
### fullselect:



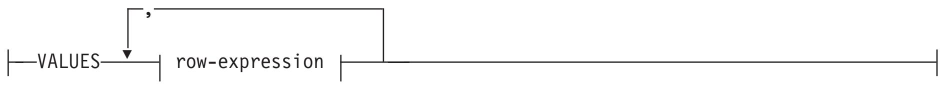
### subselect:



### select-clause:



### values-clause:



### row-expression:

```
├──┬──┬─expression─┬──────────────┬────────────────────────────────────┤
   │  └─NULL───────┘              │
   │      ┌─,────────────┐        │
   │      ▼              │        │
   └─(────┬─expression─┬─┴──)─────┘
          └─NULL───────┘
```

**Notes:**

1. When CALL is issued through the command line processor, only the identified procedures and their respective parameters are supported:

   **jar-url** Specifies the URL that contains the jar file to be installed or replaced. The only supported URL scheme is `file:`.

   **jar-id** Specifies the jar identifier in the database to be associated with the file specified by the *jar-url*.

   For example:

   ```
   db2 call sqlj.install_jar ( "file:///C:/java/jarfiles/myprocs.zip", "myprocs" )
   ```
2. The CLP version of CONNECT permits the user to change the password, using the following parameters:

   **NEW password**
   Specifies the new password that is to be assigned to the user name. Passwords can be up to 18 characters in length. The system on which the password will be changed depends on how user authentication has been set up.

   **CONFIRM password**
   A string that must be identical to the new password. This parameter is used to catch entry errors.

   **CHANGE PASSWORD**
   If this option is specified, the user is prompted for the current password, a new password, and for confirmation of the new password. Passwords are not displayed at entry.
3. When FETCH or SELECT is issued through the command line processor, decimal and floating-point numbers are displayed with the country's decimal delimiter, that is, a period (.) in the U.S., Canada, and the U.K.; a comma (,) in most other countries. However, when INSERT, UPDATE, and other SQL statements are issued through the command line processor to update tables, a period must be used as the decimal delimiter, even in countries that use a comma for that purpose.
4. When FETCH or SELECT is issued through the command line processor, null values are typically displayed as a hyphen (-). For databases

configured with DFT_SQLMATHWARN YES, expressions that result in an arithmetic error are processed as null values. Such arithmetic error nulls are displayed as a plus (+).

For example, create and populate table t1 as follows:

```
create table t1 (i1 int , i2 int);
insert into t1 values (1,1),(2,0),(3,null);
```

The statement: `select i1/i2 from t1` generates the following result:

```
1
---
  1
  +
  -
3 records selected
```

5. A new LOB option has been added to FETCH. If the LOB clause is specified, only the next row is fetched:
   - Each LOB column value is fetched into a file with the name *filename.xxx*, where *filename* is specified in the LOB clause, and *xxx* is a file extension from `001` to `999` (`001` is the first LOB column in the select list of the corresponding DECLARE CURSOR statement, `002` is the second LOB column, and `999` is the 999th column). The maximum number of LOB columns that can be fetched into files is 999.
   - Names of the files containing the data are displayed in the LOB columns.
6. When SELECT is issued through the command line processor to query tables containing LOB columns, each LOB column is truncated to 4KB in the output.
7. The command line processor displays BLOB columns in hexadecimal representation.

Change the way that the CLP displays data (when querying databases using SQL statements through the CLP) by rebinding the CLP bind files against the database being queried. For example, to display date and time in ISO format, do the following:

1. Create a text file containing the names of the CLP bind files. This file is used as the list file for binding multiple files with one BIND command. In this example the file is named `clp.lst`, and its contents are:

```
db2clpcs.bnd +
db2clprr.bnd +
db2clpur.bnd +
db2clprs.bnd +
db2clpns.bnd
```

2. Connect to the database.

3. Issue the following command:

```
db2 bind @clp.lst collection nullid datetime iso
```

For detailed information about the command line processor, see "Chapter 2. Command Line Processor (CLP)" on page 93. For more information about the syntax of SQL statements and the function provided by SQL statements, see the *SQL Reference*. For information about reading syntax diagrams, see "Appendix A. How to Read the Syntax Diagrams" on page 521.

Table 8. SQL Statements (DB2 Universal Database)

| **SQL Statement** | **Dynamic**[1] | **Command Line Processor (CLP)** | **Call Level Interface**[3] **(CLI)** |
|---|---|---|---|
| ALTER { BUFFERPOOL, NODEGROUP, TABLE, TABLESPACE, TYPE, VIEW } | X | X | X |
| BEGIN DECLARE SECTION[2] | | | |
| CALL | | X[9] | X[4] |
| CLOSE | | X | `SQLCloseCursor()`, `SQLFreeStmt()` |
| COMMENT ON | X | X | X |
| COMMIT | X | X | `SQLEndTran`, `SQLTransact()` |
| Compound SQL | | | X[4] |
| CONNECT (Type 1) | | X | `SQLBrowseConnect()`, `SQLConnect()`, `SQLDriverConnect()` |
| CONNECT (Type 2) | | X | `SQLBrowseConnect()`, `SQLConnect()`, `SQLDriverConnect()` |
| CREATE { ALIAS, BUFFERPOOL, DISTINCT TYPE, EVENT MONITOR, FUNCTION, INDEX, NODEGROUP, PROCEDURE, SCHEMA, TABLE, TABLESPACE, TRIGGER, TYPE, VIEW } | X | X | X |
| DECLARE CURSOR[2] | | X | `SQLAllocStmt()` |
| DELETE | X | X | X |
| DESCRIBE[8] | | X | `SQLColAttributes()`, `SQLDescribeCol()`, `SQLDescribParam()`[6] |
| DISCONNECT | | X | `SQLDisconnect()` |
| DROP | X | X | X |

Table 8. SQL Statements (DB2 Universal Database) (continued)

| SQL Statement | Dynamic[1] | Command Line Processor (CLP) | Call Level Interface[3] (CLI) |
|---|---|---|---|
| END DECLARE SECTION[2] | | | |
| EXECUTE | | | `SQLExecute()` |
| EXECUTE IMMEDIATE | | | `SQLExecDirect()` |
| EXPLAIN | X | X | X |
| FETCH | | X | `SQLExtendedFetch()`[7], `SQLFetch()`, `SQLFetchScroll()`[7] |
| FREE LOCATOR | | | X[4] |
| GRANT | X | X | X |
| INCLUDE[2] | | | |
| INSERT | X | X | X |
| LOCK TABLE | X | X | X |
| OPEN | | X | `SQLExecute()`, `SQLExecDirect()` |
| PREPARE | | | `SQLPrepare()` |
| RELEASE | | X | |
| RENAME TABLE | X | X | X |
| REVOKE | X | X | X |
| ROLLBACK | X | X | `SQLEndTran()`, `SQLTransact()` |
| select-statement | X | X | X |
| SELECT INTO | | | |
| SET CONNECTION | | X | `SQLSetConnection()` |
| SET CONSTRAINTS | X | X | X |
| SET CURRENT DEGREE | X | X | X |
| SET CURRENT EXPLAIN MODE | X | X | X, `SQLSetConnectAttr()` |
| SET CURRENT EXPLAIN SNAPSHOT | X | X | X, `SQLSetConnectAttr()` |
| SET CURRENT FUNCTION PATH | X | X | X |
| SET CURRENT PACKAGESET | | | |
| SET CURRENT QUERY OPTIMIZATION | X | X | X |
| SET EVENT MONITOR STATE | X | X | X |
| SET transition-variable[5] | X | X | X |

Table 8. SQL Statements (DB2 Universal Database) (continued)

| SQL Statement | Dynamic[1] | Command Line Processor (CLP) | Call Level Interface[3] (CLI) |
|---|---|---|---|
| SIGNAL SQLSTATE[5] | X | X | X |
| UPDATE | X | X | X |
| VALUES INTO | | | |
| WHENEVER[2] | | | |

**Note:**

1. You can code all statements in this list as static SQL, but only those marked with X as dynamic SQL.
2. You cannot execute this statement.
3. An X indicates that you can execute this statement using either `SQLExecDirect()` or `SQLPrepare()` and `SQLExecute()`. If there is an equivalent DB2 CLI function, the function name is listed.
4. Although this statement is not dynamic, with DB2 CLI you can specify this statement when calling either `SQLExecDirect()`, or `SQLPrepare()` and `SQLExecute()`.
5. You can only use this within CREATE TRIGGER statements.
6. You can only use the SQL DESCRIBE statement to describe output, whereas with DB2 CLI you can also describe input (using the `SQLDescribeParam()` function).
7. You can only use the SQL FETCH statement to fetch one row at a time in one direction, whereas with the DB2 CLI `SQLExtendedFetch()` and `SQLFetchScroll()` functions, you can fetch into arrays. Furthermore, you can fetch in any direction, and at any position in the result set.
8. The DESCRIBE SQL statement has a different syntax than that of the CLP DESCRIBE command. For information on the DESCRIBE SQL statement, refer to the *SQL Reference*. For information on the DESCRIBE CLP command, refer to the *Command Reference*.
9. When CALL is issued through the command line processor, only certain procedures and their respective parameters are supported (see page 513).

# Appendix A. How to Read the Syntax Diagrams

A syntax diagram shows how a command should be specified so that the operating system can correctly interpret what is typed.

Read a syntax diagram from left to right, and from top to bottom, following the horizontal line (the main path). If the line ends with an arrowhead, the command syntax is continued, and the next line starts with an arrowhead. A vertical bar marks the end of the command syntax.

When typing information from a syntax diagram, be sure to include punctuation, such as quotation marks and equal signs.

Parameters are classified as keywords or variables:

- Keywords represent constants, and are shown in uppercase letters; at the command prompt, however, keywords can be entered in upper, lower, or mixed case. A command name is an example of a keyword.
- Variables represent names or values that are supplied by the user, and are shown in lowercase letters; at the command prompt, however, variables can be entered in upper, lower, or mixed case, unless case restrictions are explicitly stated. A file name is an example of a variable.

A parameter can be a combination of a keyword and a variable.

Required parameters are displayed on the main path:

```
►►──COMMAND─required parameter──────────────────────────────────────►◄
```

Optional parameters are displayed below the main path:

```
►►──COMMAND──┬────────────────────┬─────────────────────────────────►◄
             └─optional parameter─┘
```

A parameter's default value is displayed above the path:

```
►►──COMMAND──┬────────────────────────┬─────────────────────────────►◄
             │            ┌─VALUE1─┐  │
             └─OPTPARM────┼─VALUE2─┼──┘
                          ├─VALUE3─┤
                          └─VALUE4─┘
```

## How to Read the Syntax Diagrams

A stack of parameters, with the first parameter displayed on the main path, indicates that one of the parameters must be selected:

```
►►──COMMAND──┬─required choice1─┬──────────────────────────────────────────►◄
             └─required choice2─┘
```

A stack of parameters, with the first parameter displayed below the main path, indicates that one of the parameters can be selected:

```
►►──COMMAND──┬──────────────────┬──────────────────────────────────────────►◄
             ├─optional_choice1─┤
             └─optional_choice2─┘
```

An arrow returning to the left, above the path, indicates that items can be repeated in accordance with the following conventions:

- If the arrow is uninterrupted, the item can be repeated in a list with the items separated by blank spaces:

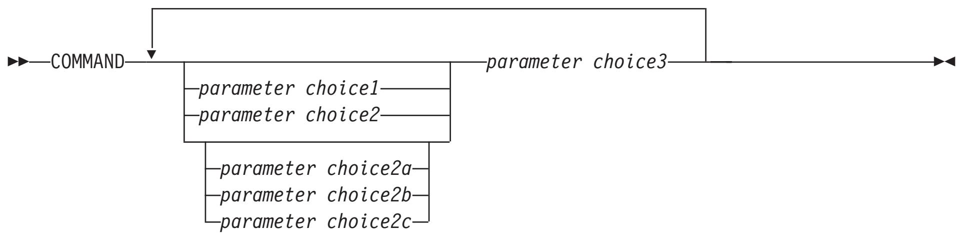
```
             ┌──────────────────────┐
             ▼                      │
►►──COMMAND────repeatable parameter─┴──────────────────────────────────────►◄
```

- If the arrow contains a comma, the item can be repeated in a list with the items separated by commas:

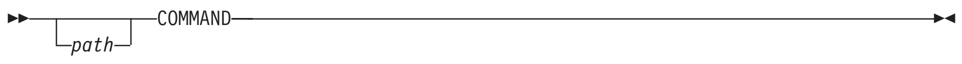
```
             ┌─,────────────────────┐
             ▼                      │
►►──COMMAND────repeatable_parameter─┴──────────────────────────────────────►◄
```

Items from parameter stacks can be repeated in accordance with the stack conventions for required and optional parameters discussed previously.

Some syntax diagrams contain parameter stacks within other parameter stacks. Items from stacks can only be repeated in accordance with the conventions discussed previously. That is, if an inner stack does not have a repeat arrow above it, but an outer stack does, only one parameter from the inner stack can be chosen and combined with any parameter from the outer stack, and that combination can be repeated. For example, the following diagram shows that one could combine parameter *choice2a* with parameter *choice2*, and then repeat that combination again (*choice2* plus *choice2a*):

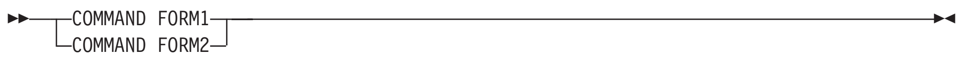


Some commands are preceded by an optional path parameter:



If this parameter is not supplied, the system searches the current directory for the command. If it cannot find the command, the system continues searching for the command in all the directories on the paths listed in the `.profile`.

Some commands have syntactical variants that are functionally equivalent:



# Appendix B. Naming Conventions

This section provides information about the conventions that apply when naming database manager objects, such as databases and tables, and authentication IDs.

- Character strings that represent names of database manager objects can contain any of the following: a-z, A-Z, 0-9, @, #, and $.
- The first character in the string must be an alphabetic character, @, #, or $; it cannot be a number or the letter sequences SYS, DBM, or IBM.
- Unless otherwise noted, names can be entered in lowercase letters; however, the database manager processes them as if they were uppercase.

  The exception to this is character strings that represent names under the systems network architecture (SNA). Many values, such as logical unit names (partner_lu and local_lu), are case sensitive. The name must be entered exactly as it appears in the SNA definitions that correspond to those terms.
- A database name or database alias is a unique character string containing from one to eight letters, numbers, or keyboard characters from the set described above.

  Databases are cataloged in the system and local database directories by their aliases in one field, and their original name in another. For most functions, the database manager uses the name entered in the alias field of the database directories. (The exceptions are CHANGE DATABASE COMMENT and CREATE DATABASE, where a directory path must be specified.)
- The name or the alias name of a table or a view is an SQL identifier that is a unique character string 1 to 128 characters in length. Column names can be 1 to 30 characters in length.

  A fully qualified table name consists of the *schema.tablename*. The schema is the unique user ID under which the table was created.
- Authentication IDs (both user IDs and group IDs) cannot exceed eight characters in length.
- Local aliases for remote nodes that are to be cataloged in the node directory cannot exceed eight characters in length.

For more information about naming conventions, see the *Administration Guide*. For more information about length limits for all DB2 identifiers, see the *SQL Reference*.

# Appendix C. How the DB2 Library Is Structured

The DB2 Universal Database library consists of SmartGuides, online help, books and sample programs in HTML format. This section describes the information that is provided, and how to access it.

To access product information online, you can use the Information Center. You can view task information, DB2 books, troubleshooting information, sample programs, and DB2 information on the Web. See “Accessing Information with the Information Center” on page 538 for details.

## Completing Tasks with SmartGuides

SmartGuides help you complete some administration tasks by taking you through each task one step at a time. SmartGuides are available through the Control Center and the Client Configuration Assistant. The following table lists the SmartGuides.

**Note:** Create Database, Index, and Configure Multisite Update SmartGuide are available for the partitioned database environment.

| SmartGuide | Helps You to... | How to Access... |
|---|---|---|
| *Add Database* | Catalog a database on a client workstation. | From the Client Configuration Assistant, click **Add**. |
| *Back up Database* | Determine, create, and schedule a backup plan. | From the Control Center, click with the right mouse button on the database you want to back up and select **Backup**->**Database using SmartGuide**. |
| *Configure Multisite Update SmartGuide* | Perform a multi-site update, a distributed transaction, or a two-phase commit. | From the Control Center, click with the right mouse button on the **Database** icon and select **Multisite Update**. |
| *Create Database* | Create a database, and perform some basic configuration tasks. | From the Control Center, click with the right mouse button on the **Databases** icon and select **Create**->**Database using SmartGuide**. |

| SmartGuide | Helps You to... | How to Access... |
|---|---|---|
| *Create Table* | Select basic data types, and create a primary key for the table. | From the Control Center, click with the right mouse button on the **Tables** icon and select **Create**->**Table using SmartGuide**. |
| *Create Table Space* | Create a new table space. | From the Control Center, click with the right mouse button on the **Table spaces** icon and select **Create**->**Table space using SmartGuide**. |
| *Index* | Advise which indexes to create and drop for all your queries. | From the Control Center, click with the right mouse button on the **Index** icon and select **Create**->**Index using SmartGuide**. |
| *Performance Configuration* | Tune the performance of a database by updating configuration parameters to match your business requirements. | From the Control Center, click with the right mouse button on the database you want to tune and select **Configure using SmartGuide**. |
| *Restore Database* | Recover a database after a failure. It helps you understand which backup to use, and which logs to replay. | From the Control Center, click with the right mouse button on the database you want to restore and select **Restore**->**Database using SmartGuide**. |

## Accessing Online Help

Online help is available with all DB2 components. The following table describes the various types of help. You can also access DB2 information through the Information Center. For information see "Accessing Information with the Information Center" on page 538.

| Type of Help | Contents | How to Access... |
|---|---|---|
| *Command Help* | Explains the syntax of commands in the command line processor. | From the command line processor in interactive mode, enter:<br><br>? *command*<br><br>where *command* is a keyword or the entire command.<br><br>For example, `? catalog` displays help for all the CATALOG commands, while `? catalog database` displays help for the CATALOG DATABASE command. |

| Type of Help | Contents | How to Access... |
|---|---|---|
| *Control Center Help*<br><br>*Client Configuration Assistant Help*<br><br>*Event Analyzer Help*<br><br>*Command Center Help* | Explains the tasks you can perform in a window or notebook. The help includes prerequisite information you need to know, and describes how to use the window or notebook controls. | From a window or notebook, click the **Help** push button or press the F1 key. |
| *Message Help* | Describes the cause of a message, and any action you should take. | From the command line processor in interactive mode, enter:<br><br>? *XXXnnnnn*<br><br>where *XXXnnnnn* is a valid message identifier.<br><br>For example, `? SQL30081` displays help about the SQL30081 message.<br><br>To view message help one screen at a time, enter:<br><br>? *XXXnnnnn* \| `more`<br><br>To save message help in a file, enter:<br><br>? *XXXnnnnn* > *filename.ext*<br><br>where *filename.ext* is the file where you want to save the message help. |
| *SQL Help* | Explains the syntax of SQL statements. | From the command line processor in interactive mode, enter:<br><br>`help` *statement*<br><br>where *statement* is an SQL statement.<br><br>For example, **help** SELECT displays help about the SELECT statement.<br>**Note:** SQL help is not available on UNIX-based platforms. |
| *SQLSTATE Help* | Explains SQL states and class codes. | From the command line processor in interactive mode, enter:<br><br>**?** *sqlstate* or **?** *class-code*<br><br>where *sqlstate* is a valid five-digit SQL state and *class-code* is the first two digits of the SQL state.<br><br>For example, `? 08003` displays help for the 08003 SQL state, while `? 08` displays help for the 08 class code. |

## DB2 Information – Hardcopy and Online

The table in this section lists the DB2 books. They are divided into two groups:

**Cross-platform books**
These books contain the common DB2 information for all platforms.

**Platform-specific books**
These books are for DB2 on a specific platform. For example, there are separate *Quick Beginnings* books for DB2 on OS/2, on Windows NT, and on the UNIX-based platforms.

**Cross-platform sample programs in HTML**
These samples are the HTML version of the sample programs that are installed with the SDK. They are for informational purposes and do not replace the actual programs.

Most books are available in HTML and PostScript format, or you can choose to order a hardcopy from IBM. The exceptions are noted in the table.

On OS/2 and Windows platforms, HTML documentation files can be installed under the doc\html subdirectory. Depending on the language of your system, some files may be in that language, and the remainder are in English.

On UNIX platforms, you can install multiple language versions of the HTML documentation files under the doc/%L/html subdirectories. Any documentation that is not available in a national language is shown in English.

You can obtain DB2 books and access information in a variety of different ways:

| | |
|---|---|
| **View** | See "Viewing Online Information" on page 537. |
| **Search** | See "Searching Online Information" on page 540. |
| **Print** | See "Printing the PostScript Books" on page 540. |
| **Order** | See "Ordering the Printed Books" on page 541. |

| Name | Description | Form Number<br><br>File Name for Online Book | HTML Directory |
|---|---|---|---|
| | **Cross-Platform Books** | | |

| Name | Description | Form Number<br><br>File Name for Online Book | HTML Directory |
|---|---|---|---|
| *Administration Guide* | *Administration Guide, Design and Implementation* contains information required to design, implement, and maintain a database. It also describes database access using the Control Center(whether local or in a client/server environment), auditing, database recovery, distributed database support, and high availability.<br><br>*Administration Guide, Performance* contains information that focuses on the database environment, such as application performance evaluation and tuning.<br><br>You can order both volumes of the *Administration Guide* in the English language in North America using the form number SBOF-8922. | Volume 1<br>SC09-2839<br>db2d1x60<br><br>Volume 2<br>SC09-2840<br>db2d2x60 | db2d0 |
| *Administrative API Reference* | Describes the DB2 application programming interfaces (APIs) and data structures you can use to manage your databases. Explains how to call APIs from your applications. | SC09-2841<br><br>db2b0x60 | db2b0 |
| *Application Building Guide* | Provides environment setup information and step-by-step instructions about how to compile, link, and run DB2 applications on Windows, OS/2, and UNIX-based platforms.<br><br>This book combines the *Building Applications* books for the OS/2, Windows, and UNIX-based environments. | SC09-2842<br><br>db2axx60 | db2ax |
| *APPC, CPI-C and SNA Sense Codes* | Provides general information about APPC, CPI-C, and SNA sense codes that you may encounter when using DB2 Universal Database products.<br>**Note:** Available in HTML format only. | No form number<br><br>db2apx60 | db2ap |

| Name | Description | Form Number<br><br>File Name for Online Book | HTML Directory |
|---|---|---|---|
| *Application Development Guide* | Explains how to develop applications that access DB2 databases using embedded SQL or JDBC, how to write stored procedures, user-defined types, user-defined functions, and how to use triggers. It also discusses programming techniques and performance considerations.<br><br>This book was formerly known as the *Embedded SQL Programming Guide*. | SC09-2845<br><br>db2a0x60 | db2a0 |
| *CLI Guide and Reference* | Explains how to develop applications that access DB2 databases using the DB2 Call Level Interface, a callable SQL interface that is compatible with the Microsoft ODBC specification. | SC09-2843<br><br>db2l0x60 | db2l0 |
| *Command Reference* | Explains how to use the command line processor, and describes the DB2 commands you can use to manage your database. | SC09-2844<br><br>db2n0x60 | db2n0 |
| *Data Movement Utilities Guide and Reference* | Explains how to use the Load, Import, Export, Autoloader, and Data Propogation utilities to work with the data in the database. | SC09-2858<br><br>db2dmx60 | db2dm |
| *DB2 Connect Personal Edition Quick Beginnings* | Provides planning, installing, and configuring information for DB2 Connect Personal Edition. | GC09-2830<br><br>db2c1x60 | db2c1 |
| *DB2 Connect User's Guide* | Provides concepts, programming and general usage information about the DB2 Connect products. | SC09-2838<br><br>db2c0x60 | db2c0 |
| *Connectivity Supplement* | Provides setup and reference information on how to use DB2 for AS/400, DB2 for OS/390, DB2 for MVS, or DB2 for VM as DRDA application requesters with DB2 Universal Database servers, and on how to use DRDA application servers with DB2 Connect application requesters.<br>**Note:** Available in HTML and PostScript formats only. | No form number<br><br>db2h1x60 | db2h1 |
| *Glossary* | Provides a comprehensive list of all DB2 terms and definitions.<br>**Note:** Available in HTML format only. | No form number<br><br>db2t0x50 | db2t0 |

| **Name** | **Description** | **Form Number**<br><br>**File Name for Online Book** | **HTML Directory** |
|---|---|---|---|
| *Installation and Configuration Supplement* | Guides you through the planning, installation, and set up of platform-specific DB2 clients. This supplement contains information on binding, setting up client and server communications, DB2 GUI tools, DRDA AS, distributed installation, and the configuration of distributed requests and access methods to heterogeneous data sources. | GC09-2857<br><br>db2iyx60 | db2iy |
| *Message Reference* | Lists messages and codes issued by DB2, and describes the actions you should take. | GC09-2846<br><br>db2m0x60 | db2m0 |
| *Replication Guide and Reference* | Provides planning, configuration, administration, and usage information for the IBM Replication tools supplied with DB2. | SC26-9642<br><br>db2e0x60 | db2e0 |
| *SQL Getting Started* | Introduces SQL concepts, and provides examples for many constructs and tasks. | SC09-2856<br><br>db2y0x60 | db2y0 |
| *SQL Reference*, Volume 1 and Volume 2 | Describes SQL syntax, semantics, and the rules of the language. Also includes information about release-to-release incompatibilities, product limits, and catalog views.<br><br>You can order both volumes of the *SQL Reference* in the English language in North America with the form number SBOF-8923. | SBOF-8923<br><br>Volume 1 db2s1x60<br><br>Volume 2 db2s2x60 | db2s0 |
| *System Monitor Guide and Reference* | Describes how to collect different kinds of information about databases and the database manager. Explains how to use the information to understand database activity, improve performance, and determine the cause of problems. | SC09-2849<br><br>db2f0x60 | db2f0 |
| *Troubleshooting Guide* | Helps you determine the source of errors, recover from problems, and use diagnostic tools in consultation with DB2 Customer Service. | S10J-8169 | db2p0 |

| Name | Description | Form Number<br><br>File Name for Online Book | HTML Directory |
|---|---|---|---|
| *What's New* | Describes the new features, functions, and enhancements in DB2 Universal Database, Version 6.0, including information about Java-based tools. | SC09-2851<br><br>db2q0x60 | db2q0 |
| | **Platform-Specific Books** | | |
| *Administering Satellites Guide and Reference* | Provides planning, configuration, administration, and usage information for satellites. | GC09-2821<br><br>db2dsx60 | db2ds |
| *DB2 Personal Edition Quick Beginnings* | Provides planning, installation, migration, and configuration information for DB2 Universal Database Personal Edition on the OS/2, Windows 95, and Windows NT operating systems. | GC09-2831<br><br>db2i1x60 | db2i1 |
| *DB2 for OS/2 Quick Beginnings* | Provides planning, installation, migration, and configuration information for DB2 Universal Database on the OS/2 operating system. Also contains installing and setup information for many supported clients. | GC09-2834<br><br>db2i2x60 | db2i2 |
| *DB2 for UNIX Quick Beginnings* | Provides planning, installation, migration, and configuration information for DB2 Universal Database on UNIX-based platforms. Also contains installing and setup information for many supported clients. | GC09-2836<br><br>db2ixx60 | db2ix |
| *DB2 for Windows NT Quick Beginnings* | Provides planning, installation, migration, and configuration information for DB2 Universal Database on the Windows NT operating system. Also contains installing and setup information for many supported clients. | GC09-2835<br><br>db2i6x60 | db2i6 |
| *DB2 Enterprise - Extended Edition for UNIX Quick Beginnings* | Provides planning, installation, and configuration information for DB2 Enterprise - Extended Edition for UNIX. Also contains installing and setup information for many supported clients. | GC09-2832<br><br>db2v3x60 | db2v3 |

| Name | Description | Form Number<br><br>File Name for Online Book | HTML Directory |
|---|---|---|---|
| *DB2 Enterprise - Extended Edition for Windows NT Quick Beginnings* | Provides planning, installation, and configuration information for DB2 Enterprise - Extended Edition for Windows NT. Also contains installing and setup information for many supported clients. | GC09-2833<br><br>db2v6x60 | db2v6 |
| *DB2 Connect Enterprise Edition for OS/2 and Windows NT Quick Beginnings* | Provides planning, migration, installation, and configuration information for DB2 Connect Enterprise Edition on the OS/2 and Windows NT operating systems. Also contains installation and setup information for many supported clients.<br><br>This book was formerly part of the *DB2 Connect Enterprise Edition Quick Beginnings*. | GC09-2828<br><br>db2c6x60 | db2c6 |
| *DB2 Connect Enterprise Edition for UNIX Quick Beginnings* | Provides planning, migration, installation, configuration, and usage information for DB2 Connect Enterprise Edition in UNIX-based platforms. Also contains installation and setup information for many supported clients.<br><br>This book was formerly part of the *DB2 Connect Enterprise Edition Quick Beginnings*. | GC09-2829<br><br>db2cyx60 | db2cy |
| *DB2 Data Links Manager for AIX Quick Beginnings* | Provides planning, installation, configuration, and task information for DB2 Data Links Manager for AIX. | GC09-2837<br><br>db2z0x60 | db2z0 |
| *DB2 Data Links Manager for Windows NT Quick Beginnings* | Provides planning, installation, configuration, and task information for DB2 Data Links Manager for Windows NT. | GC09-2827<br><br>db2z6x60 | db2z6 |
| *DB2 Query Patroller Administration Guide* | Provides administration information on DB2 Query Patrol. | SC09-2859<br><br>db2dwx60 | db2dw |
| *DB2 Query Patroller Installation Guide* | Provides installation information on DB2 Query Patrol. | GC09-2860<br><br>db2iwx60 | db2iw |
| *DB2 Query Patroller User's Guide* | Describes how to use the tools and functions of the DB2 Query Patrol. | SC09-2861<br><br>db2wwx60 | db2ww |

| Name | Description | Form Number<br><br>File Name for Online Book | HTML Directory |
|---|---|---|---|
| | **Cross-Platform Sample Programs in HTML** | | |
| Sample programs in HTML | Provides the sample programs in HTML format for the programming languages on all platforms supported by DB2 for informational purposes (not all samples are available in all languages). Only available when the SDK is installed.<br><br>See *Application Building Guide* for more information on the actual programs.<br>**Note:** Available in HTML format only. | No form number | db2hs/c<br>db2hs/cli<br>db2hs/clp<br>db2hs/cpp<br>db2hs/cobol<br>db2hs/cobol_mf<br>db2hs/fortran<br>db2hs/java<br>db2hs/rexx |

**Notes:**

1. The character in the sixth position of the file name indicates the language of a book. For example, the file name `db2d0e60` indicates that the *Administration Guide* is in English. The following letters are used in the file names to indicate the language of a book:

| **Language** | **Identifier** |
|---|---|
| Brazilian Portuguese | b |
| Bulgarian | u |
| Czech | x |
| Danish | d |
| Dutch | q |
| English | e |
| Finnish | y |
| French | f |
| German | g |
| Greek | a |
| Hungarian | h |
| Italian | i |
| Japanese | j |
| Korean | k |
| Norwegian | n |
| Polish | p |
| Portuguese | v |
| Russian | r |
| Simp. Chinese | c |
| Slovenian | l |
| Spanish | z |

| | |
|---|---|
| Swedish | s |
| Trad. Chinese | t |
| Turkish | m |

2. For late breaking information that could not be included in the DB2 books:
   - On UNIX-based platforms, see the Release.Notes file. This file is located in the DB2DIR/Readme/%L directory, where `%L` is the locale name and DB2DIR is:
     - /usr/lpp/db2_06_01 on AIX
     - /opt/IBMdb2/V6.1 on HP-UX, Solaris, SCO UnixWare 7, and Silicon Graphics IRIX
     - /usr/IBMdb2/V6.1 on Linux.
   - On other platforms, see the RELEASE.TXT file. This file is located in the directory where the product is installed.
   - Under Windows `Start` menu

## Viewing Online Information

The manuals included with this product are in Hypertext Markup Language (HTML) softcopy format. Softcopy format enables you to search or browse the information, and provides hypertext links to related information. It also makes it easier to share the library across your site.

You can view the online books or sample programs with any browser that conforms to HTML Version 3.2 specifications.

To view online books or sample programs on all platforms other than SCO UnixWare 7:

- If you are running DB2 administration tools, use the Information Center. See "Accessing Information with the Information Center" on page 538 for details.
- Select the `Open Page` menu item of your Web browser. The page you open contains descriptions of and links to DB2 information:
  - On UNIX-based platforms, open the following page:

    ```
    file:/INSTHOME/sqllib/doc/%L/html/index.htm
    ```

    where *%L* is the locale name.
  - On other platforms, open the following page:

    ```
    sqllib\doc\html\index.htm
    ```

    The path is located on the drive where DB2 is installed.

If you have not installed the Information Center, you can open the page by double-clicking on the **DB2 Online Books** icon. Depending on the system you are using, the icon is in the main product folder or the Windows Start menu.

To view online books or sample programs on the SCO UnixWare 7:

- DB2 Universal Database for SCO UnixWare 7 uses the native SCOhelp utility to search the DB2 information. You can access SCOhelp by the following methods:
  - entering the ″scohelp″ command on the command line,
  - selecting the `Help` menu in the Control Panel of the CDE desktop or
  - selecting `Help` in the Root menu of the Panorama desktop

  For more information on SCOhelp, refer to the *Installation and Configuration Supplement*.

## Accessing Information with the Information Center

The Information Center provides quick access to DB2 product information. The Information Center is available on all platforms on which the DB2 administration tools are available.

Depending on your system, you can access the Information Center from the:

- Main product folder
- Toolbar in the Control Center
- Windows Start menu
- Help menu of the Control Center

The Information Center provides the following kinds of information. Click the appropriate tab to look at the information:

| | |
|---|---|
| **Tasks** | Lists tasks you can perform using DB2. |
| **Reference** | Lists DB2 reference information, such as keywords, commands, and APIs. |
| **Books** | Lists DB2 books. |
| **Troubleshooting** | Lists categories of error messages and their recovery actions. |
| **Sample Programs** | Lists sample programs that come with the DB2 Software Developer’s Kit. If the Software Developer’s Kit is not installed, this tab is not displayed. |
| **Web** | Lists DB2 information on the World Wide |

Web. To access this information, you must have a connection to the Web from your system.

When you select an item in one of the lists, the Information Center launches a viewer to display the information. The viewer might be the system help viewer, an editor, or a Web browser, depending on the kind of information you select.

The Information Center provides some search capabilities, so you can look for specific topics, and filter capabilities to limit the scope of your searches.

For a full text search, click the Search button of the Information Center follow the *Search DB2 Books* link in each HTML file.

The HTML search server is usually started automatically. If a search in the HTML information does not work, you may have to start the search server by double-clicking its icon on the Windows or OS/2 desktop.

Refer to the release notes if you experience any other problems when searching the HTML information.

**Note:** Search function is not available in the Linux and Silicon Graphics environments.

## Setting Up a Document Server

By default, the DB2 information is installed on your local system. This means that each person who needs access to the DB2 information must install the same files. To have the DB2 information stored in a single location, use the following instructions:

1. Copy all files and subdirectories from \sqllib\doc\html on your local system to a Web server. Each book has its own subdirectory containing all the necessary HTML and GIF files that make up the book. Ensure that the directory structure remains the same.
2. Configure the Web server to look for the files in the new location. For information, see the NetQuestion Appendix in *Installation and Configuration Supplement*.
3. If you are using the Java version of the Information Center, you can specify a base URL for all HTML files. You should use the URL for the list of books.
4. Once you are able to view the book files, you should bookmark commonly viewed topics. Among those, you will probably want to bookmark the following pages:

- List of books
- Tables of contents of frequently used books
- Frequently referenced articles, such as the *ALTER TABLE* topic
- The Search form

For information about setting up a search, see the NetQuestion Appendix in *Installation and Configuration Supplement* book.

## Searching Online Information

To search for information in the HTML books, you can do the following:

- Click on **Search the DB2 Books** at the bottom of any page in the HTML books. Use the search form to find a specific topic. This function is not available in the Linux or Silicon Graphics IRIX environments.
- Click on **Index** at the bottom of any page in an HTML book. Use the index to find a specific topic in the book.
- Display the table of contents or index of the HTML book, and then use the find function of the Web browser to find a specific topic in the book.
- Use the bookmark function of the Web browser to quickly return to a specific topic.
- Use the search function of the Information Center to find specific topics. See "Accessing Information with the Information Center" on page 538 for details.

## Printing the PostScript Books

If you prefer to have printed copies of the manuals, you can decompress and print PostScript versions. For the file name of each book in the library, see the table in "DB2 Information – Hardcopy and Online" on page 530. Specify the full path name for the file you intend to print.

On OS/2 and Windows platforms:

1. Copy the compressed PostScript files to a hard drive on your system. The files have a file extension of .exe and are located in the x:\doc\*language*\books\ps directory, where x: is the letter representing the CD-ROM drive and *language* is the two-character country code that represents your language (for example, EN for English).
2. Decompress the file that corresponds to the book that you want. Each compressed book is a self-extracting executable file. To decompress the

book, simply run it as you would run any other executable program. The result from this step is a printable PostScript file with a file extension of .ps.
3. Ensure that your default printer is a PostScript printer capable of printing Level 1 (or equivalent) files.
4. Enter the following command from a command line:
   ```
   print filename.ps
   ```

On UNIX-based platforms:
1. Mount the CD-ROM. Refer to your *Quick Beginnings* manual for the procedures to mount the CD-ROM.
2. Change to /cdrom/doc/%L/ps directory on the CD-ROM, where */cdrom* is the mount point of the CD-ROM and *%L* is the name of the desired locale. The manuals will be installed in the previously-mentioned directory with file names ending with .ps.Z.
3. Decompress and print the manual you require using the following command:
   - For AIX:
     ```
     zcat filename | qprt -P PSPrinter_queue
     ```
   - For HP-UX, Solaris, or SCO UnixWare 7:
     ```
     zcat filename | lp -d PSPrinter_queue
     ```
   - For Linux:
     ```
     zcat filename | lpr -P PSPrinter_queue
     ```
   - For Silicon Graphics IRIX:
     ```
     zcat < filename | lp -d PSPrinter_queue
     ```

   where *filename* is the full path name and extension of the compressed PostScript file and *PSprinter_queue* is the name of the PostScript printer queue.

   For example, to print the English version of *DB2 for UNIX Quick Beginnings* on AIX, you can use the following command:
   ```
   zcat /cdrom/doc/en/ps/db2ixe60.ps.Z || qprt -P ps1
   ```

## Ordering the Printed Books

You can order the printed DB2 manuals either as a set or individually. There are three sets of books available. The form number for the entire set of DB2 books is SBOF-8926-00. The form number for the books listed under the heading ″Cross-Platform Books″ is SBOF-8924-00.

**Note:** These form numbers only apply if you are ordering books that are printed in the English language in North America.

You can also order books individually by the form number listed in "DB2 Information – Hardcopy and Online" on page 530. To order printed versions, contact your IBM authorized dealer or marketing representative, or phone `1-800-879-2755` in the United States or `1-800-IBM-4YOU` in Canada.

# Appendix D. Notices

Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's licensed program may be used. Any functionally equivalent product, program or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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

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

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

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

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

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

## Trademarks

The following terms are trademarks or registered trademarks of the IBM Corporation in the United States and / or other countries:

ACF / VTAM
ADSTAR
AISPO
AIX
AIXwindows
AnyNet
APPN
AS / 400
CICS
C Set++
C / 370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
Extended Services
FFST
First Failure Support Technology
IBM
IMS
LAN Distance
MVS / ESA
MVS / XA
OS / 400
OS / 390
OS / 2
PowerPC
QMF
RACF
RISC System / 6000
SP
SQL / DS
SQL / 400
S / 370
System / 370
System / 390
SystemView
VisualAge
VM / ESA
VSE / ESA
VTAM
WIN-OS / 2

## Trademarks of Other Companies

The following terms are trademarks or registered trademarks of the companies listed:

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

HP-UX is a trademark of Hewlett-Packard.

Java, HotJava, Solaris, Solstice, and Sun are trademarks of Sun Microsystems, Inc.

# Index

# Contacting IBM

This section lists ways you can get more information from IBM.

If you have a technical problem, please take the time to review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. Depending on the nature of your problem or concern, this guide will suggest information you can gather to help us to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

**Telephone**

If you live in the U.S.A., call one of the following numbers:

- 1-800-237-5511 to learn about available service options.
- 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

For information on how to contact IBM outside of the United States, see Appendix A of the IBM Software Support Handbook. You can access this document by accessing the following page:

`http://www.ibm.com/support/`

then performing a search using the keyword "handbook".

Note that in some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

**World Wide Web**
http://www.software.ibm.com/data/
http://www.software.ibm.com/data/db2/library/

The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more. The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information. (Note that this information may be in English only.)

**Anonymous FTP Sites**
ftp.software.ibm.com

Log on as anonymous. In the directory /ps/products/db2, you can find demos, fixes, information, and tools concerning DB2 and many related products.

**Internet Newsgroups**
comp.databases.ibm-db2, bit.listserv.db2-l
These newsgroups are available for users to discuss their experiences with DB2 products.

**CompuServe**
**GO IBMDB2** to access the IBM DB2 Family forums
All DB2 products are supported through these forums.

To find out about the IBM Professional Certification Program for DB2 Universal Database, go to http://www.software.ibm.com/data/db2/db2tech/db2cert.html

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC09-2844-00