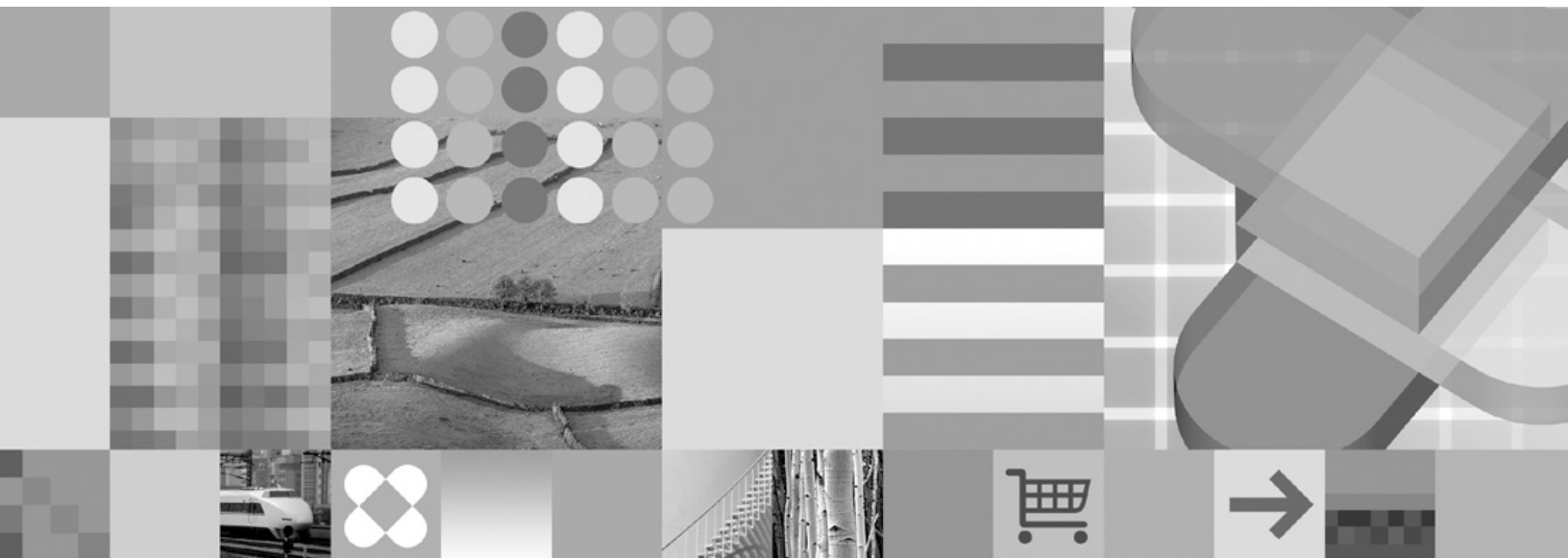




Command Reference



Command Reference

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

Edition Notice

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

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

© Copyright International Business Machines Corporation 1993, 2006. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About This Book vii

Who Should Use this Book vii

How this Book is Structured vii

Chapter 1. System Commands 1

How the command descriptions are organized 1

dasauto - Autostart DB2 administration server 3

dasCRT - Create a DB2 administration server 4

dasdrop - Remove a DB2 administration server 5

dasmigr - Migrate the DB2 administration server 6

dasupdt - Update DAS 8

db2_deinstall - Uninstall DB2 products or features 10

db2_install - Install DB2 product 11

db2admin - DB2 administration server 13

db2adutl - Managing DB2 objects within TSM 15

db2advis - DB2 design advisor 23

db2audit - Audit facility administrator tool 29

db2batch - Benchmark tool 34

db2bfd - Bind file description tool 43

db2ca - Start the Configuration Assistant 44

db2cap - CLI/ODBC static package binding tool 45

db2cat - System catalog analysis 47

db2cc - Start control center 49

db2cfexp - Connectivity configuration export tool 51

db2cfimp - Connectivity configuration import tool 53

db2chglbpath - Modify the embedded runtime 54

library search path 54

db2chgpath - Change embedded runtime path 56

db2ckbkp - Check backup 57

db2ckmig - Database pre-migration tool 61

db2ckrst - Check incremental restore image 63

sequence 63

db2cli - DB2 interactive CLI 65

db2cmd - Open DB2 command window 66

db2dart - Database analysis and reporting tool 67

db2daslevel - Show DAS level 71

db2dclgn - Declaration generator 72

db2diag - db2diag.log analysis tool 75

db2drdat - DRDA trace 86

db2drvmp - DB2 database drive map 88

db2empfa - Enable multipage file allocation 90

db2eva - Event analyzer 91

db2evmon - Event monitor productivity tool 93

db2evtbl - Generate event monitor target table 95

definitions 95

db2exfmt - Explain table format 97

db2exmig - Migrate explain tables 99

db2expln - SQL and XQuery Explain 100

db2extsec - Set permissions for DB2 objects 105

db2flsn - Find log sequence number 106

db2fm - DB2 fault monitor 108

db2fs - First steps 110

db2gcf - Control DB2 instance 111

db2gov - DB2 governor 113

db2govlg - DB2 governor log query 115

db2gpmap - Get distribution map 116

db2hc - Start health center 117

db2iauto - Auto-start instance 118

db2iclus - Microsoft cluster server 119

db2icrt - Create instance 122

db2idrop - Remove instance 125

db2ilist - List instances 127

db2imigr - Migrate instance 128

db2inidb - Initialize a mirrored database 130

db2inspf - Format inspect results 132

db2isetup - Start instance creation interface 133

db2iuupd - Update instances 135

db2jdbcbind - DB2 JDBC package binder 138

db2ldcfg - Configure LDAP environment 140

db2level - Show DB2 service level 141

db2licm - License management tool 142

db2listvolumes - Display GUIDs for all disk 144

volumes 144

db2logsforrwd - List logs required for rollforward 145

recovery 145

db2look - DB2 statistics and DDL extraction tool 146

db2ls - List installed DB2 products and features 155

db2move - Database movement tool 157

db2mqdsn - MQ listener 165

db2mscs - Set up Windows failover utility 169

db2mtrk - Memory tracker 173

db2nchg - Change database partition server 176

configuration 176

db2ncrt - Add database partition server to an 178

instance 178

db2ndrop - Drop database partition server from an 180

instance 180

db2osconf - Utility for kernel parameter values 181

db2pd - Monitor and troubleshoot DB2 database 184

db2pdcfg - Configure DB2 database for problem 223

determination behavior 223

db2perfc - Reset database performance values 226

db2perfi - Performance counters registration utility 228

db2perfr - Performance monitor registration tool 229

db2rbind - Rebind all packages 230

db2_recon_aid - Reconcile multiple tables 232

db2relocatedb - Relocate database 235

db2rfpen - Reset rollforward pending state 240

db2rspgn - Response file generator (Windows) 241

db2sAMPL - Create sample database 242

db2set - DB2 profile registry 245

db2setup - Install DB2 248

db2sql92 - SQL92 compliant SQL statement 249

processor 249

db2sqljbind - SQLJ profile binder 252

db2sqljcustomize - SQLJ profile customizer 259

db2sqljprint - SQLJ profile printer 270

db2start - Start DB2 271

db2stop - Stop DB2 272

db2support - Problem analysis and environment 273

db2swtch - Switch default DB2 copy	278
db2sync - Start DB2 synchronizer	279
db2systray - Start DB2 system tray	280
db2tapemgr - Manage log files on tape	282
db2tbst - Get table space state	285
db2trc - Trace	286
db2uiddl - Prepare unique index conversion to V5 semantics	290
db2undgp - Revoke execute privilege	291
db2unins - Uninstall DB2 database product	292
db2untag - Release container tag	294
db2xprt - Format trap file	295
doce_deinstall - Uninstall DB2 Information Center	296
doce_install - Install DB2 Information Center	297
disable_MQFunctions	299
enable_MQFunctions	301
installFixPack - Update installed DB2 products	303
setup - Install DB2.	305
sqlj - SQLJ translator	307

Chapter 2. Command Line Processor (CLP) 311

db2 - Command line processor invocation	311
Command line processor options	312
Command line processor return codes	320
Command line processor features	321
Command Line Processor Help	326
Invoking message help from the command line processor	326
Invoking command help from the command line processor	326

Chapter 3. CLP Commands 327

DB2 CLP Commands	327
ACTIVATE DATABASE	330
ADD CONTACT	333
ADD CONTACTGROUP	335
ADD DBPARTITIONNUM	336
ADD XMLSCHEMA DOCUMENT	339
ARCHIVE LOG	341
ATTACH	344
AUTOCONFIGURE	346
BACKUP DATABASE	349
BIND	355
CATALOG DATABASE	372
CATALOG DCS DATABASE	375
CATALOG LDAP DATABASE	377
CATALOG LDAP NODE	381
CATALOG LOCAL NODE	382
CATALOG NAMED PIPE NODE	384
CATALOG ODBC DATA SOURCE	386
CATALOG TCPIP/TCPIP4/TCPIP6 NODE	387
CHANGE DATABASE COMMENT	390
CHANGE ISOLATION LEVEL	392
COMPLETE XMLSCHEMA	394
CREATE DATABASE	395
CREATE TOOLS CATALOG	408
DEACTIVATE DATABASE	411
DECOMPOSE XML DOCUMENT	413
DEREGISTER	415

DESCRIBE	417
DETACH	423
DROP CONTACT	424
DROP CONTACTGROUP	425
DROP DATABASE	426
DROP DBPARTITIONNUM VERIFY	428
DROP TOOLS CATALOG	429
ECHO	431
EDIT	432
EXPORT	433
FORCE APPLICATION	439
GET ADMIN CONFIGURATION	441
GET ALERT CONFIGURATION	443
GET AUTHORIZATIONS	449
GET CLI CONFIGURATION	451
GET CONNECTION STATE	453
GET CONTACTGROUP	454
GET CONTACTGROUPS	455
GET CONTACTS	456
GET DATABASE CONFIGURATION	457
GET DATABASE MANAGER CONFIGURATION	463
GET DATABASE MANAGER MONITOR SWITCHES	468
GET DESCRIPTION FOR HEALTH INDICATOR	471
GET HEALTH NOTIFICATION CONTACT LIST	473
GET HEALTH SNAPSHOT	474
GET INSTANCE	477
GET MONITOR SWITCHES	478
GET RECOMMENDATIONS FOR HEALTH INDICATOR	481
GET ROUTINE	485
GET SNAPSHOT	487
HELP	492
HISTORY	493
IMPORT	494
INITIALIZE TAPE	511
INSPECT	512
LIST ACTIVE DATABASES	518
LIST APPLICATIONS	520
LIST COMMAND OPTIONS	522
LIST DATABASE DIRECTORY	523
LIST DATABASE PARTITION GROUPS	526
LIST DBPARTITIONNUMS	528
LIST DCS APPLICATIONS	529
LIST DCS DIRECTORY	531
LIST DRDA INDOUBT TRANSACTIONS	533
LIST HISTORY	535
LIST INDOUBT TRANSACTIONS	538
LIST NODE DIRECTORY	541
LIST ODBC DATA SOURCES	544
LIST PACKAGES/TABLES	545
LIST TABLESPACE CONTAINERS	548
LIST TABLESPACES	550
LIST UTILITIES	555
LOAD	557
LOAD QUERY	576
MIGRATE DATABASE	579
PING	581
PRECOMPILE	583
PRUNE HISTORY/LOGFILE	607
PUT ROUTINE	609

QUERY CLIENT	611
QUIESCE	612
QUIESCE TABLESPACES FOR TABLE	615
QUIT	618
REBIND	619
RECONCILE	623
RECOVER DATABASE	627
REDISTRIBUTE DATABASE PARTITION GROUP	633
REFRESH LDAP	636
REGISTER	637
REGISTER XMLSCHEMA	640
REGISTER XSROBJECT	642
REORG INDEXES/TABLE	644
REORGCHK	654
RESET ADMIN CONFIGURATION	663
RESET ALERT CONFIGURATION	665
RESET DATABASE CONFIGURATION	667
RESET DATABASE MANAGER CONFIGURATION	669
RESET MONITOR	671
RESTART DATABASE	673
RESTORE DATABASE	675
REWIND TAPE	690
ROLLFORWARD DATABASE	691
RUNCMD	701
RUNSTATS	702
SET CLIENT	716
SET RUNTIME DEGREE	719
SET TABLESPACE CONTAINERS	721
SET TAPE POSITION	723
SET UTIL_IMPACT_PRIORITY	724
SET WRITE	726
START DATABASE MANAGER	728
START HADR	734
STOP DATABASE MANAGER	736
STOP HADR	739
TAKEOVER HADR	741
TERMINATE	744
UNCATALOG DATABASE	745
UNCATALOG DCS DATABASE	747
UNCATALOG LDAP DATABASE	749
UNCATALOG LDAP NODE	751
UNCATALOG NODE	752
UNCATALOG ODBC DATA SOURCE	753
UNQUIESCE	754
UPDATE ADMIN CONFIGURATION	756
UPDATE ALERT CONFIGURATION	758
UPDATE ALTERNATE SERVER FOR DATABASE	762
UPDATE ALTERNATE SERVER FOR LDAP DATABASE	764
UPDATE CLI CONFIGURATION	766
UPDATE COMMAND OPTIONS	768

UPDATE CONTACT	770
UPDATE CONTACTGROUP	771
UPDATE DATABASE CONFIGURATION	772
UPDATE DATABASE MANAGER CONFIGURATION	775
UPDATE HEALTH NOTIFICATION CONTACT LIST	777
UPDATE HISTORY	778
UPDATE LDAP NODE	780
UPDATE MONITOR SWITCHES	782

Chapter 4. Using command line SQL statements and XQuery statements . . . 785

Appendix A. How to read the syntax diagrams 793

Appendix B. Naming conventions . . . 797

Appendix C. File type modifiers . . . 799

File type modifiers for the load utility	799
File type modifiers for the import utility	810
File type modifiers for the export utility	821
Delimiter restrictions for moving data	826

Appendix D. DB2 Database technical information 829

Overview of the DB2 technical information	829
Documentation feedback	829
DB2 technical library in hardcopy or PDF format	830
Ordering printed DB2 books	832
Displaying SQL state help from the command line processor	833
Accessing different versions of the DB2 Information Center	833
Displaying topics in your preferred language in the DB2 Information Center	834
Updating the DB2 Information Center installed on your computer or intranet server	835
DB2 tutorials	836
DB2 troubleshooting information	837
Terms and Conditions	837

Appendix E. Notices 839

Trademarks	841
----------------------	-----

Appendix F. Contacting IBM 843

Index 845

About This Book

This book provides information about the use of system commands and the IBM DB2 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.

Appendix C

Describes the file type modifiers for the load, import and export utilities.

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.

Notes:

1. Slashes (/) in directory paths are specific to UNIX based systems, and are equivalent to back slashes (\) in directory paths on Windows® operating systems.
2. The term Windows normally refers to all supported versions of Microsoft® Windows. Supported versions include those versions based on Windows NT and those based on Windows 9x. Specific references to "Windows NT-based operating systems" may occur when the function in question is supported on Windows NT® 4, Windows 2000, Windows .NET and Windows XP but not on Windows 9x. If there is a function that is specific to a particular version of Windows, the valid version or versions of the operating system will be noted.

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-database-partition system, the scope is that single database partition only. In a multi-database-partition system, it is the collection of all logical database partitions defined in the database partition 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.

Command syntax:

A syntax diagram shows how a command should be specified so that the operating system can correctly interpret what is typed. For more information about syntax diagrams, see Appendix A, "How to read the syntax diagrams," on page 793.

Command parameters:

System Commands

A description of the parameters available to the command.

Usage notes:

Other information.

Related reference:

A cross-reference to related information.

dasauto - Autostart DB2 administration server

Enables or disables autostarting of the DB2[®] administration server.

This command is available on Linux[®] and UNIX systems only. It is located in the DB2DIR/das/adm directory, where DB2DIR is the location where the current version of the DB2 database product is installed.

Authorization:

dasadm

Required connection:

None

Command syntax:

```

▶▶ dasauto [-h] [-on] [-off]

```

Command parameters:

- h/-?** Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.
- on** Enables autostarting of the DB2 administration server. The next time the system is restarted, the DB2 administration server will be started automatically.
- off** Disables autostarting of the DB2 administration server. The next time the system is restarted, the DB2 administration server will not be started automatically.

Related tasks:

- “Configuring the DB2 administration server (DAS)” in *Administration Guide: Implementation*
- “Creating a DB2 administration server (DAS)” in *Administration Guide: Implementation*
- “Starting and stopping the DB2 administration server (DAS)” in *Administration Guide: Implementation*

dasprt - Create a DB2 administration server

The DB2 administration server (DAS) provides support services for DB2 tools such as the Control Center and the Configuration Assistant. If a system does not have a DAS, you can use this command to manually generate it. The **dasprt** command is located in the DB2DIR/instance directory, where DB2DIR is the location where the current version of the DB2 database product is installed.

This command is available on Linux and UNIX based systems only. On Windows operating systems, you can use the **db2admin create** command for the same purpose.

Authorization:

Root authority.

Required connection:

None.

Command syntax:

```
▶▶ dasprt -u DASuser [-d] ▶▶
```

Command parameters:

-u *DASuser*

DASuser is the user ID under which the DAS will be created. The DAS will be created under the `/home/DASuser/das` directory.

-d Enters debug mode, for use with DB2 Service.

Related tasks:

- “Configuring the DB2 administration server (DAS)” in *Administration Guide: Implementation*
- “Creating a DB2 administration server (DAS)” in *Administration Guide: Implementation*
- “Starting and stopping the DB2 administration server (DAS)” in *Administration Guide: Implementation*

dasdrop - Remove a DB2 administration server

On Linux and UNIX based systems only, removes the DB2 Administration Server (DAS). The Administration Server provides support services for DB2 tools such as the Control Center and the Configuration Assistant. On Windows operating systems, you can use the **db2admin drop** command for the same purpose.

Authorization:

Root authority.

Required connection:

None.

Command syntax:

▶▶—dasdrop [-d] ▶▶

Command parameters:

-d Enters debug mode, for use with DB2 Service.

Usage notes:

- The **dasdrop** command is located in the DB2DIR/instance directory, where DB2DIR is the location where the current version of the DB2 database product is installed.

Related tasks:

- “Removing the DB2 administration server (DAS)” in *Administration Guide: Implementation*

dasmigr - Migrate the DB2 administration server

Migrates the DB2 Administration Server (DAS) on the system from a previous version of DB2 database system (supported for migration to the current version of DB2 database system) to the current version of DB2 database system at the DB2 database level related to the path where the **dasmigr** is issued.

To move the DAS from one DB2 database system installation location to another within the same version of DB2 database system, the **dasupdt** command should be used. A DAS at a previous version of a DB2 database system can not be used to administer instances in the current version of DB2 database system.

On Linux and UNIX systems, this utility is located in the DB2DIR/instance directory. On Windows operating systems, it is located in the DB2DIR\bin directory. DB2DIR represents the installation location where the current version of the DB2 database system is installed.

Authorization:

Root access on UNIX operating systems or Local Administrator authority on Windows operating systems.

Required connection:

None.

Command syntax:

For Linux and UNIX systems

```
▶▶—dasmigr [ -d ]
```

For Windows operating systems

```
▶▶—dasmigr [ -h ] [ -p—path override ]
```

Command parameters:

For Linux and UNIX systems

-d Enters debug mode, for use with DB2 Service.

For Windows operating systems

-h Displays usage information.

-p*path override*

Indicates that the DAS profile should be moved as well. *path override* is a user specified path to be used instead of the default DAS profile path.

Examples:

On Linux and UNIX systems

```
DB2DIR/instance/dasmig
```


dasmigr - Migrate the DB2 Administration Server

On Windows operating systems

```
dasmigr db2as dasusr1
```

Related tasks:

- “Configuring the DB2 administration server (DAS)” in *Administration Guide: Implementation*
- “Migrating the DB2 Administration Server (DAS)” in *Migration Guide*

Related reference:

- “dasupdt - Update DAS” on page 8

dasupdt - Update DAS

On Linux and UNIX-based systems, updates the DB2 Administration Server (DAS) if the related DB2 database system installation is updated. This utility is located in the DB2DIR/instance directory, where DB2DIR is the location where the current version of the DB2 database product is installed. You can also use this utility to move the DAS from one installation location to another if both are at the same version of DB2 database system.

On Windows operating systems, updates the DAS from one DB2 copy to another within the same DB2 database release. To move a DAS from an older release use the **dasmigr** command. The DAS will be moved to the DB2 copy that the **dasupdt** command is executed from. This utility is located in the DB2DIR\bin directory, where DB2DIR is the location where the current version of the DB2 database product is installed.

Authorization:

Root access on Linux and UNIX-based systems or Local Administrator authority on Windows operating systems.

Required connection:

None

Command syntax:

For Linux and UNIX-based systems

```
▶▶ dasupdt [-d] [-D] [-h] [-?]
```

For Windows operating systems

```
▶▶ dasmigr [-h] [-p path override]
```

Command parameters:

For Linux and UNIX-based systems

- d Sets the debug mode, which is used for problem analysis.
- D Moves the DAS from a higher code level on one path to a lower code level installed on another path.
- h or -? Displays usage information.

For Windows operating systems

- h Displays usage information.
- p *path override* Indicates that the DAS profile should be moved as well. *path override* is a user specified path to be used instead of the default DAS profile path.

Examples:

If you have applied a Fix Pack to a DB2 installation path which has the DAS it is related to, the following command, issued from the installation path, will update the DAS to the applied Fix Pack:

```
dasupdt
```

If a DAS is running in one DB2 installation path and you want to move the DAS to another installation path at a lower level (but the two installation paths are at the same version of DB2 database system), issue the following command from the installation path at the lower level:

```
dasupdt -D
```

Related tasks:

- “Configuring the DB2 administration server (DAS)” in *Administration Guide: Implementation*
- “Updating a DB2 administration server (DAS) configuration for discovery” in *Administration Guide: Implementation*
- “Updating the DB2 administration server (DAS) on UNIX” in *Administration Guide: Implementation*

db2_deinstall - Uninstall DB2 products or features

Uninstalls all the installed DB2 products or features that are in the same install path as the **db2_deinstall** tool. It is only available on Linux and UNIX systems.

The **db2_deinstall** command is located at `DB2DIR/install`, where `DB2DIR` is the location where the current version of the DB2 database product is installed. The **db2_deinstall** command can be used to uninstall only the DB2 products related to the installation path.

Authorization:

Root

Required Connection:

None.

Command syntax:

```
▶▶ db2_deinstall [-F feature-name] [-a] [-l log-file] [-t trace-file]
▶▶ [-h] [-?]
```

Command parameters:

- F *feature-name***
Specifies the removal of one feature. To indicate uninstallation of multiple features, specify this parameter multiple times. For example, `-F feature1 -F feature2`.
- a**
Removes all installed DB2 products in the current location.
- l *log-file***
Specifies the log file. The default log file is `/tmp/db2_deinstall.log$$`, where `$$` is the process ID.
- t *trace-file***
Turns on the debug mode. The debug information is written to the file name specified as *trace-file*.
- h/-?**
Displays usage information.

Examples:

- To uninstall all the DB2 database products that are installed in a location (`DB2DIR`), issue the `db2_deinstall` located at `DB2DIR/install` directory:
`DB2DIR/install/db2_deinstall -a`

Related reference:

- “`db2ls` - List installed DB2 products and features” on page 155

db2_install - Install DB2 product

Installs all features of a DB2 product to the given path. This command is available only on Linux and UNIX-based systems.

Authorization:

Root

Required Connection:

None.

Command syntax:

```

▶▶ db2_install [ -b install-path ] [ -p product ] [ -c image-location ]
               [ -n ] [ -L language ] [ -l log-file ] [ -t trace-file ] [ -h ]
               [ -? ]

```

Command parameters:

-b *install-path*

Specifies the path where the DB2 product is to be installed. *install-path* must be a full path name and its maximum length is limited to 128 characters. This parameter is mandatory when the **-n** parameter is specified.

-p *productID*

Specifies the DB2 product to be installed. *productID* does not require DB2 as a prefix. This parameter is case insensitive and is mandatory when the **-n** parameter is specified.

-c *image-location*

Specifies the product image location. To indicate multiple image locations, specify this parameter multiple times. For example, **-c CD1 -c CD2**. This parameter is only mandatory if the **-n** parameter is specified, your install requires more than one CD, and your images are not set up for automatic discovery. Otherwise, you are prompted for the location of the next CD at the time it is needed. For details on automatic discovery associated with multiple installation images, see Multiple CD installation (Linux and UNIX).

-n Specifies non-interactive mode.

-L *language*

Specifies national language support. You can install a non-English version of a DB2 product. However, you must run this command from the product CD, not the National Language pack CD. By default, English is always installed, therefore, English does not need to be specified. When more than one language is required, this parameter is mandatory. To indicate multiple languages, specify this parameter multiple times. For example, to install both French and German, specify **-L FR -L DE**. This parameter is case insensitive.

db2_install - Install DB2 product

- l** *log-file*
Specifies the log file. The default log file is `/tmp/db2_install.log$$`, where `$$` is the process ID.
- t** *trace-file*
Turns on the debug mode. The debug information is written to the file name specified as *trace-file*.
- h/-?** Displays usage information.

Examples:

- To install from an image in `/mnt/cdrom`, and to be prompted for all needed input, issue: To install DB2 Enterprise Server Edition from an image in `/mnt/cdrom`, issue:

```
cd /mnt/cdrom
./db2_install
```
- To install DB2 Enterprise Server Edition to `/db2/newlevel`, from an image in `/mnt/cdrom`, non-interactively in English, issue:

```
cd /mnt/cdrom
./db2_install -p ese -b /db2/newlevel -n
```

Related tasks:

- “Installing a DB2 product using the `db2_install` or `doce_install` command (Linux and UNIX)” in *Installation and Configuration Supplement*

db2admin - DB2 administration server

This utility is used to manage the DB2 Administration Server (DAS). If no parameters are specified, and the DAS exists, this command returns the name of the DAS.

On Linux and UNIX based systems, the executable file for the **db2admin** command can be found in the DASHOME/das/bin directory, where DASHOME is the home directory of the DAS user. On Windows operating systems, the **db2admin** executable is found under the DB2PATH\bin directory where DB2PATH is the location where the DB2 copy is installed.

Authorization:

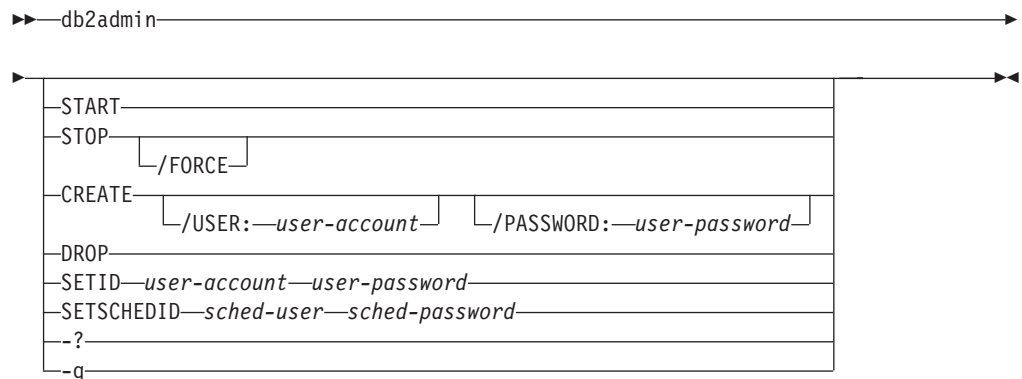
dasadm on UNIX operating systems but not associated with a 64-bit instance.

Local administrator on Windows operating systems.

Required connection:

None

Command syntax:



Command parameters:

START

Start the DAS.

STOP /FORCE

Stop the DAS. The force option is used to force the DAS to stop, regardless of whether or not it is in the process of servicing any requests.

CREATE /USER: *user-account* /PASSWORD: *user-password*

Create the DAS. If a user name and password are specified, the DAS 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 DAS functions can be accessed. To create a DAS on UNIX operating systems, use the **dasCRT** command.

DROP Deletes the DAS. To drop a DAS on UNIX operating systems you must use the **dasdrop** command.

db2admin - DB2 Administration Server

SETID *user-account/user-password*

Establishes or modifies the user account associated with the DAS.

SETSCHEDID *sched-user/sched-password*

Establishes the logon account used by the scheduler to connect to the tools catalog database. Only required if the scheduler is enabled and the tools catalog database is remote to the DAS. For more information about the scheduler, see the *Administration Guide*.

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

-q Run the **db2admin** command in quiet mode. No messages will be displayed when the command is run. This option can be combined with any of the other command options.

Related tasks:

- “Tools catalog database and DB2 administration server (DAS) scheduler setup and configuration” in *Administration Guide: Implementation*

Related reference:

- “dasCRT - Create a DB2 administration server ” on page 4
- “dasDROP - Remove a DB2 administration server” on page 5

db2adutl - Managing DB2 objects within TSM

Allows users to query, extract, verify, and delete backup images, logs, and load copy images saved using Tivoli Storage Manager (TSM). Also allows users to grant and revoke access to objects on a TSM server.

On UNIX operating systems, this utility is located in the `sql1lib/adsm` directory. On Windows operating systems, it is located in `sql1lib\bin`.

Authorization:

None

Required connection:

None

Command syntax:

```
db2adutl db2-object-options access-control-options
```

db2-object-options:

```

QUERY-options
EXTRACT-options
DELETE-options
VERIFY-options
COMPRLIB—decompression-library
COMPROPTS—decompression-options
VERBOSE
DATABASE—database_name
DB
DBPARTITIONNUM—db-partition-number
PASSWORD—password
NODENAME—node_name
OWNER—owner
WITHOUT PROMPTING

```

QUERY-options:

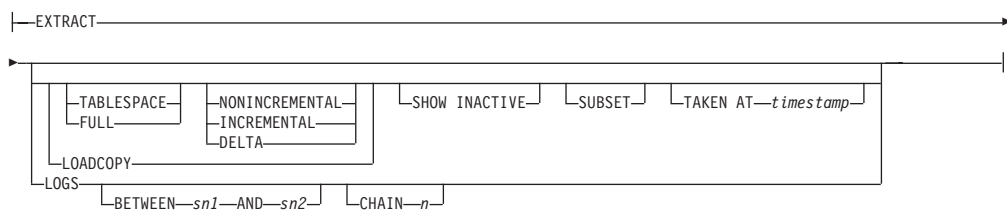
```

QUERY
TABLESPACE
FULL
LOADCOPY
LOGS
NONINCREMENTAL
INCREMENTAL
DELTA
SHOW INACTIVE
BETWEEN—sn1—AND—sn2
CHAIN—n

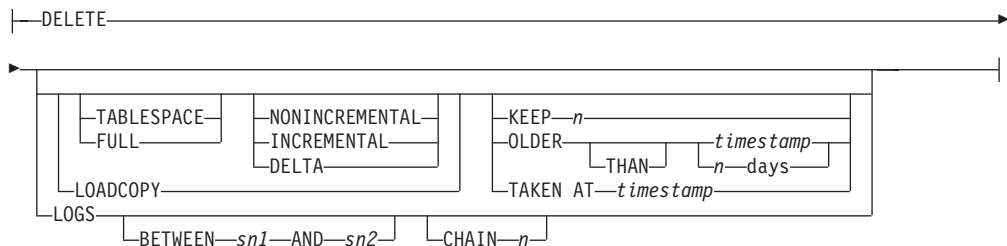
```

db2adutl - Managing DB2 objects within TSM

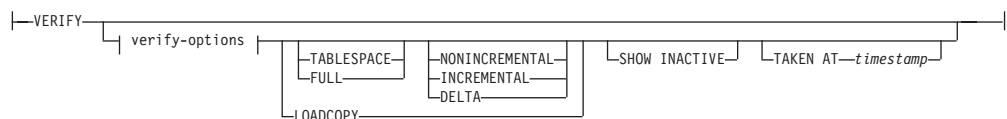
EXTRACT-options:



DELETE-options:



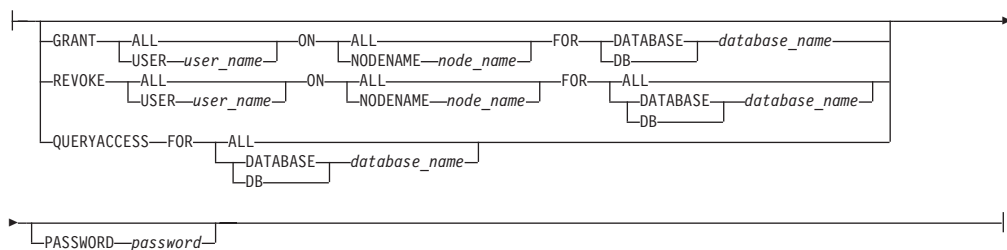
VERIFY-options:



verify-options:



access-control-options:



Command parameters:

QUERY

Queries the TSM server for DB2 objects.

EXTRACT

Copies DB2 objects from the TSM server to the current directory on the local machine.

DELETE

Either deactivates backup objects or deletes log archives on the TSM server.

VERIFY

Performs consistency checking on the backup copy that is on the server. This parameter causes the entire backup image to be transferred over the network.

ALL Displays all available information.

CHECK

Displays results of checkbits and checksums.

DMS Displays information from headers of DMS table space data pages.

HEADER

Displays the media header information.

HEADERONLY

Displays the same information as HEADER but only reads the 4 K media header information from the beginning of the image. It does not validate the image.

LFH Displays the log file header (LFH) data.

OBJECT

Displays detailed information from the object headers.

PAGECOUNT

Displays the number of pages of each object type found in the image.

SGF Displays the automatic storage paths in the image.

SGFONLY

Displays only the automatic storage paths in the image but does not validate the image.

TABLESPACES

Displays the table space details, including container information, for the table spaces in the image.

TABLESPACESONLY

Displays the same information as TABLESPACES but does not validate the image.

TABLESPACE

Includes only table space backup images.

FULL Includes only full database backup images.

NONINCREMENTAL

Includes only non-incremental backup images.

INCREMENTAL

Includes only incremental backup images.

DELTA

Includes only incremental delta backup images.

db2adutl - Managing DB2 objects within TSM

LOADCOPY

Includes only load copy images.

LOGS Includes only log archive images

BETWEEN *sn1* AND *sn2*

Specifies that the logs between log sequence number 1 and log sequence number 2 are to be used.

CHAIN *n*

Specifies the chain ID of the logs to be used.

SHOW INACTIVE

Includes backup objects that have been deactivated.

SUBSET

Extracts pages from an image to a file. To extract pages, you will need an input and an output file. The default input file is called extractPage.in. You can override the default input file name by setting the DB2LISTFILE environment variable to a full path. The format of the input file is as follows:

For SMS table spaces:

```
S <tbspID> <objID> <objType> <startPage> <numPages>
```

Notes:

1. <startPage> is an object page number that is object-relative.

For DMS table spaces:

```
D <tbspID> <objType> <startPage> <numPages>
```

Notes:

1. <objType> is only needed if verifying DMS load copy images.
2. <startPage> is an object page number that is pool-relative.

For log files:

```
L <log num> <startPos> <numPages>
```

For other data (for example, initial data):

```
0 <objType> <startPos> <numBytes>
```

The default output file is extractPage.out. You can override the default output file name by setting the DB2EXTRACTFILE environment variable to a full path.

TAKEN AT *timestamp*

Specifies a backup image by its time stamp.

KEEP *n*

Deactivates all objects of the specified type except for the most recent *n* by time stamp.

OLDER THAN *timestamp* or *n* days

Specifies that objects with a time stamp earlier than *timestamp* or *n* days will be deactivated.

COMPRLIB *decompression-library*

Indicates the name of the library to be used to perform the decompression. The name must be a fully qualified path referring to a file on the server. If this parameter is not specified, DB2 will attempt to use the library stored

in the image. If the backup was not compressed, the value of this parameter will be ignored. If the specified library cannot be loaded, the operation will fail.

COMPROPTS *decompression-options*

Describes a block of binary data that will be passed to the initialization routine in the decompression library. DB2 will pass this string directly from the client to the server, so any issues of byte reversal or code page conversion will have to be handled by the decompression library. If the first character of the data block is '@', the remainder of the data will be interpreted by DB2 as the name of a file residing on the server. DB2 will then replace the contents of the data block with the contents of this file and will pass this new value to the initialization routine instead. The maximum length for this string is 1024 bytes.

DATABASE *database_name*

Considers only those objects associated with the specified database name.

DBPARTITIONNUM *db-partition-number*

Considers only those objects created by the specified database partition number.

PASSWORD *password*

Specifies the TSM client password for this node, if required. If a database is specified and the password is not provided, the value specified for the *tsm_password* database configuration parameter is passed to TSM; otherwise, no password is used.

NODENAME *node_name*

Considers only those images associated with a specific TSM node name.

OWNER *owner*

Considers only those objects created by the specified owner.

WITHOUT PROMPTING

The user is not prompted for verification before objects are deleted.

VERBOSE

Displays additional file information.

GRANT ALL / USER *user_name*

Adds access rights to the TSM files on the current TSM node to all users or to the users specified. Granting access to users gives them access for all current and future files related to the database specified.

REVOKE ALL / USER *user_name*

Removes access rights to the TSM files on the current TSM node from all users or to the users specified.

QUERYACCESS

Retrieves the current access list. A list of users and TSM nodes is displayed.

ON ALL / NODENAME *node_name*

Specifies the TSM node for which access rights will be changed.

FOR ALL / DATABASE *database_name*

Specifies the database to be considered.

Examples:

1. The following is sample output from the command `db2 backup database rawsamp1 use tsm`

db2adutl - Managing DB2 objects within TSM

Backup successful. The timestamp for this backup is : 20031209184503

The following is sample output from the command db2adutl query issued following the backup operation:

Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.

1 Time: 20031209184403, Oldest log: S0000050.LOG, Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.

No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE BACKUP information.

No DELTA DATABASE BACKUP images found for RAWSAMPL

Retrieving TABLESPACE BACKUP information.

No TABLESPACE BACKUP images found for RAWSAMPL

Retrieving INCREMENTAL TABLESPACE BACKUP information.

No INCREMENTAL TABLESPACE BACKUP images found for RAWSAMPL

Retrieving DELTA TABLESPACE BACKUP information.

No DELTA TABLESPACE BACKUP images found for RAWSAMPL

Retrieving LOCAL COPY information.

No LOCAL COPY images found for RAWSAMPL

Retrieving log archive information.

Log file: S0000050.LOG, Chain Num: 0, DB Partition Number: 0,

Taken at 2003-12-09-18.46.13

Log file: S0000051.LOG, Chain Num: 0, DB Partition Number: 0,

Taken at 2003-12-09-18.46.43

Log file: S0000052.LOG, Chain Num: 0, DB Partition Number: 0,

Taken at 2003-12-09-18.47.12

Log file: S0000053.LOG, Chain Num: 0, DB Partition Number: 0,

Taken at 2003-12-09-18.50.14

Log file: S0000054.LOG, Chain Num: 0, DB Partition Number: 0,

Taken at 2003-12-09-18.50.56

Log file: S0000055.LOG, Chain Num: 0, DB Partition Number: 0,

Taken at 2003-12-09-18.52.39

2. The following is sample output from the command db2adutl delete full taken at 20031209184503 db rawsampl

Query for database RAWSAMPL

Retrieving FULL DATABASE BACKUP information.

Taken at: 20031209184503 DB Partition Number: 0 Sessions: 1

Do you want to delete this file (Y/N)? y

Are you sure (Y/N)? y

Retrieving INCREMENTAL DATABASE BACKUP information.

No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE BACKUP information.

No DELTA DATABASE BACKUP images found for RAWSAMPL

The following is sample output from the command db2adutl query issued following the operation that deleted the full backup image. Note the timestamp for the backup image.

Query for database RAWSAMPL

db2adutl - Managing DB2 objects within TSM

Retrieving FULL DATABASE BACKUP information.

1 Time: 20031209184403, Oldest log: S0000050.LOG, Sessions: 1

Retrieving INCREMENTAL DATABASE BACKUP information.

No INCREMENTAL DATABASE BACKUP images found for RAWSAMPL

Retrieving DELTA DATABASE BACKUP information.

No DELTA DATABASE BACKUP images found for RAWSAMPL

Retrieving TABLESPACE BACKUP information.

No TABLESPACE BACKUP images found for RAWSAMPL

Retrieving INCREMENTAL TABLESPACE BACKUP information.

No INCREMENTAL TABLESPACE BACKUP images found for RAWSAMPL

Retrieving DELTA TABLESPACE BACKUP information.

No DELTA TABLESPACE BACKUP images found for RAWSAMPL

Retrieving LOCAL COPY information.

No LOCAL COPY images found for RAWSAMPL

Retrieving log archive information.

Log file: S0000050.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at 2003-12-09-18.46.13

Log file: S0000051.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at 2003-12-09-18.46.43

Log file: S0000052.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at 2003-12-09-18.47.12

Log file: S0000053.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at 2003-12-09-18.50.14

Log file: S0000054.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at 2003-12-09-18.50.56

Log file: S0000055.LOG, Chain Num: 0, DB Partition Number: 0,
Taken at 2003-12-09-18.52.39

3. The following is sample output from the command `db2adutl queryaccess for all`

Node	User	Database Name	type
bar2	jchisan	sample	B
<all>	<all>	test	B

Access Types: B – Backup images L – Logs A - both

Usage Notes:

One parameter from each group below can be used to restrict what backup images types are included in the operation:

Granularity:

- FULL - include only database backup images.
- TABLESPACE - include only table space backup images.

Cumulativeness:

- NONINCREMENTAL - include only non-incremental backup images.
- INCREMENTAL - include only incremental backup images.
- DELTA - include only incremental delta backup images.

Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.

db2adutl - Managing DB2 objects within TSM

Related concepts:

- “Cross-node recovery with the db2adutl command and the logarchopt1 and vendoropt database configuration parameters” in *Data Recovery and High Availability Guide and Reference*

db2advis - DB2 design advisor

The DB2 Design Advisor advises users on the creation of materialized query tables (MQTs) and indexes, the repartitioning of tables, the conversion to multidimensional clustering (MDC) tables, and the deletion of unused objects. The recommendations are based on one or more SQL statements provided by the user. 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 Design Advisor outputs a DDL CLP script that includes CREATE INDEX, CREATE SUMMARY TABLE (MQT), and CREATE TABLE statements to create the recommended objects.

Structured type columns are not considered when this command is executed.

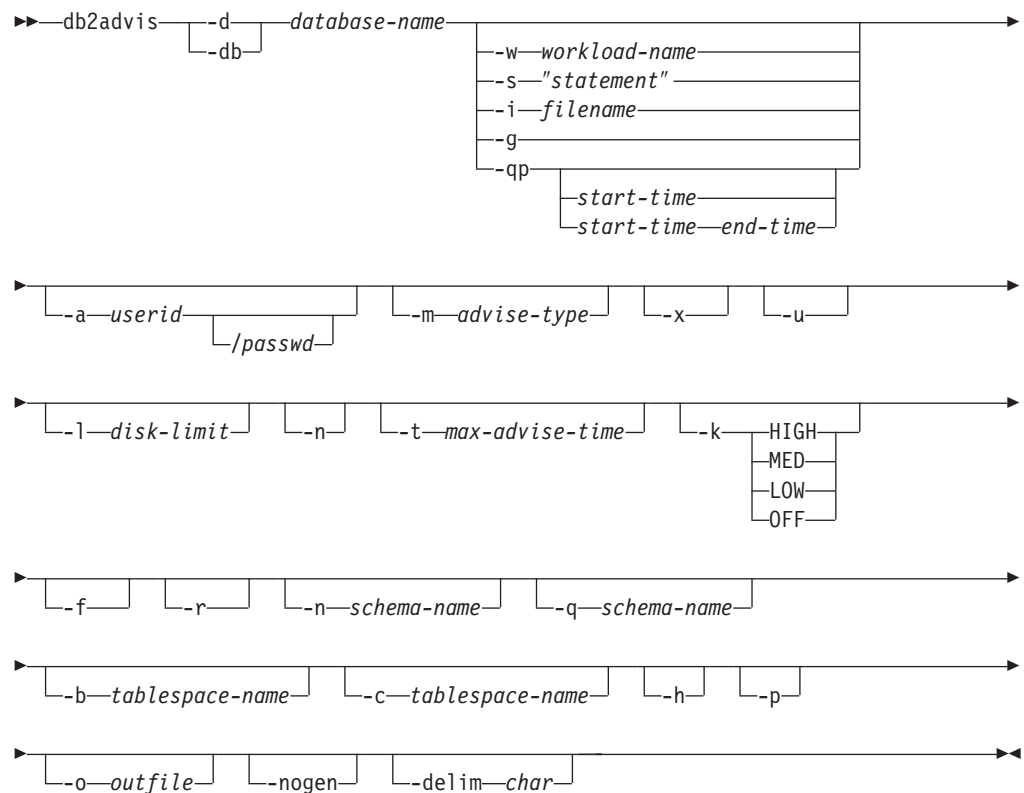
Authorization:

Read access to the database. Read and write access to the explain tables. If materialized query tables (MQTs) are used, you must have CREATE TABLE authorization, and read and write access to the MQTs.

Required connection:

None. This command establishes a database connection.

Command syntax:



Command parameters:

db2advis - DB2 Design Advisor

- 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 to be assessed and have indexes suggested by the Design Advisor. This name is used in the ADVISE_WORKLOAD table. This option cannot be specified with the -g, -i, qp, or -s options.
- s** *"statement"*
Specifies the text of a single SQL statement to be assessed and have indexes suggested by the Design Advisor. The statement must be enclosed by double quotation marks. This option cannot be specified with the -g, -i, -qp, or -w options.
- i** *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 be 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. This option cannot be specified with the -g, -s, -qp, or -w options.
- g** Specifies the retrieval of the SQL statements from a dynamic SQL snapshot. If combined with the -p command parameter, the SQL statements are kept in the ADVISE\_WORKLOAD table. This option cannot be specified with the -i, -s, -qp, or -w options.
- qp** Specifies that the workload is coming from Query Patroller. The start-time and end-time options are timestamps used to check against the time\_completed field of the DB2QP.TRACK\_QUERY\_INFO table. If no start-time and end-time timestamps are given, all rows are give "D" (for done) in the completion\_status column of the table. If only start-time is given, the rows returned are those with TIME\_COMPLETED greater than or equal to the start-time value. In addition, if the end-time value is given, the rows returned are also restricted by TIME\_COMPLETED less than or equal to the end-time value. This option cannot be used with the -w, -s, -i, or -g options.
- a** *userid/passwd*  
Name and password used to connect to the database. The slash (/) must be included if a password is specified. A password should not be specified if the -x option is specified.
- m** *advise-type*  
Specifies the type of recommendation the advisor will return. Any combination of I, M, C, and P (in upper or lower case) can be specified. For example, **db2advis -m PC** will recommend partitioning and MDC tables. If -m P or -m M are used in a partitioned database environment, the advise\_partition table is populated with the final partition recommendation. The choice of possible values are:
- I** Recommends new indexes. This is the default.

- M** Recommends new materialized query tables (MQTs) and indexes on the MQTs. In partitioned database environments, partitioning on MQTs is also recommended.
- C** Recommendation to convert standard tables to multidimensional clustering (MDC) tables; or, to create a clustering index on the tables.
- P** Recommends the repartitioning of existing tables.
- x** Specifies that the password will be read from the terminal or through user input.
- u** Specifies that the advisor will consider the recommendation of deferred MQTs. Incremental MQTs will not be recommended. When this option is specified, comments in the DDL CLP script indicate which of the MQTs could be converted to immediate MQTs. If immediate MQTs are recommended in a partitioned database environment, the default distribution key is the implied unique key for the MQT.
- l *disk-limit***  
Specifies the number of megabytes available for all recommended indexes and materialized views in the existing schema. Specify -1 to use the maximum possible size. The default value is 20% of the total database size.
- t *max-advise-time***  
Specifies the maximum allowable time, in minutes, to complete the operation. If no value is specified for this option, the operation will continue until it is completed. To specify an unlimited time enter a value of zero. The default is zero.
- k** Specifies to what degree the workload will be compressed. Compression is done to allow the advisor to reduce the complexity of the advisor's execution while achieving similar results to those the advisor could provide when the full workload is considered. HIGH indicates the advisor will concentrate on a small subset of the workload. MED indicates the advisor will concentrate on a medium-sized subset of the workload. LOW indicates the advisor will concentrate on a larger subset of the workload. OFF indicates that no compression will occur. The default is MED.
- f** Drops previously existing simulated catalog tables.
- r** Specifies that detailed statistics should be used for the virtual MQTs and for the partitioning selection. If this option is not specified, the default is to use optimizer statistics for MQTs. Although the detailed statistics might be more accurate, the time to derive them will be significant and will cause the **db2advise** execution time to be greater. The **-r** command parameter uses sampling to obtain relevant statistics for MQTs and partitioning. For MQTs, when the sample query either fails or returns no rows, the optimizer estimates are used.
- n *schema-name***  
Specifies the qualifying name of simulation catalog tables, and the qualifier for the new indexes and MQTs. The default schema name is the caller's user ID, except for catalog simulation tables where the default schema name is SYSTOOLS. The default is for new indexes to inherit the schema name of the index's base.
- q *schema-name***  
Specifies the qualifying name of unqualified names in the workload. It

## db2advis - DB2 Design Advisor

serves as the schema name to use for CURRENT SCHEMA when **db2advis** executes. The default schema name is the user ID of the person executing the command.

**-b** *tablespace-name*

Specifies the name of a table space in which new MQTs will be created. If not specified, the advisor will select the table spaces from the set of table spaces that exist.

**-c** *tablespace-name*

Specifies the name of a table space (where the table space can be of any type, for example, use a file name or directory) in which to create the simulation catalog tables. This table space must only be created on the catalog database partition group. The default is USERSPACE1.

It is recommended that the user create the table space employed for the simulation instead of using the default USERSPACE1. In addition, the ALTER TABLESPACE DROPPED TABLE RECOVERY OFF statement should be run on this table space to improve the performance of the db2advis utility. When the utility completes, turn the history back on for the table space. In a partitioned database environment, this option is required as USERSPACE1 is usually created across all partition groups.

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

**-p** Keeps the plans that were generated while running the tool in the explain tables. The **-p** command parameter causes the workload for **-qp** and **-g** to be saved in the ADVISE\_WORKLOAD table and saves the workload query plans that use the final recommendation in the explain tables.

**-o** *outfile*

Saves the script to create the recommended objects in *outfile*.

**-nogen**

Indicates that generated columns are not to be included in multidimensional clustering recommendations.

**-delim** *char*

Indicates the statement delimiter character *<char>* in a workload file input. Default is ';'.

### Examples:

1. In the following example, the utility connects to database PROTOTYPE, and recommends indexes for table ADDRESSES 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')"
```

2. In the following example, the utility connects to database PROTOTYPE, and recommends indexes that will not exceed 53MB for queries in table ADVISE\_WORKLOAD. The 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
```

3. In the following example, the input file 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) FROM EMPLOYEE
GROUP BY WORKDEPT ORDER BY WORKDEPT;
```

The utility connects to database SAMPLE, 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
```

4. In the following example, MQTs are created in table space SPACE1 and the simulation table space is SPACE2. The qualifying name for unqualified names in the workload is SCHEMA1, and the schema name in which the new MQTs will be recommended is SCHEMA2. The workload compression being used is HIGH and the disk space is unlimited. Sample statistics are used for the MQTs. Issuing the following command will recommend MQTs and, in a partitioned database environment, indexes and partitioning will also be recommended.

```
db2advis -d prototype -w production -l -l -m M -b space1 -c space2 -k
HIGH -q schema1 -n schema2 -r
```

To get the recommended MQTs, as well as indexes, partitioning and MDCs on both MQT and base tables, issue the command specifying a value of IMCP for the -m option as follows:

```
db2advis -d prototype -w production -l -l -m IMCP -b space1 -c space2 -k
HIGH -q schema1 -n schema2 -r
```

#### Usage notes:

Because these features must be set up before you can run the DDL CLP script, database partitioning, multidimensional clustering, and clustered index recommendations are commented out of the DDL CLP script that is returned. It is up to you to transform your tables into the recommended DDL. One example of doing this is to use the ALTER TABLE stored procedure but there are restrictions associated with it in the same way the RENAME command is restricted.

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

1. Issue

```
db2 reset monitor for database <database-alias>
```

Wait for an appropriate interval of time.

2. Issue

```
db2advis -g <other-options>
```

If the -p parameter is used with the -g parameter, the dynamic SQL statements obtained will be placed in the ADVISE\_WORKLOAD table with a generated workload name that contains a timestamp.

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.

Any db2advis error information can also be found in the db2diag.log.

## db2adviz - DB2 Design Advisor

When the advisor begins running, the ADVISE\_INSTANCE table will contain a row that identifies the advisor. The main advisor row is identified by the START\_TIME showing when the advisor began its run. This row's STATUS is "STARTED".

If issuing the **db2adviz** command results in an error saying "Cannot insert into DB2ADVIS\_INSTANCE", you will need to bind db2adviz.bnd and run the **db2adviz** command with the -l option. The bind operation can be performed by issuing db2 bind db2adviz.bnd blocking all grant public.

When the advisor is completed, you can check the associated row with the appropriate START\_TIME in the ADVISE\_INSTANCE table. If STATUS is "COMPLETED", the advisor executed successfully. If STATUS is still "STARTED" and there is no db2adviz process running, the advisor has terminated prematurely. If STATUS has an "EX", you are also shown an "SQLCODE" to determine how the advisor failed.

If the *-l disk-limit* option is not specified, you must have at least one of *sysadm*, *sysctrl*, *sysmaint*, or *sysmon* authority to determine the maximum database size using the *get\_dbsize\_info* stored procedure.

### Related concepts:

- "The Design Advisor" in *Performance Guide*

---

## 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.

When working in a partitioned database environment, many of the auditable events occur at the database partition at which the user is connected (the coordinator partition) or at the catalog partition (if they are not the same database partition). The implication of this is that audit records can be generated by more than one database partition. Part of each audit record contains information on the coordinator partition and originating database partition identifiers.

The audit log (db2audit.log) and the audit configuration file (db2audit.cfg) are located in the instance's security subdirectory. At the time you create an instance, read/write permissions are set on these files, where possible, by the operating system. By default, the permissions are read/write for the instance owner only. It is recommended that you do not change these permissions.

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.

### **Authorization:**

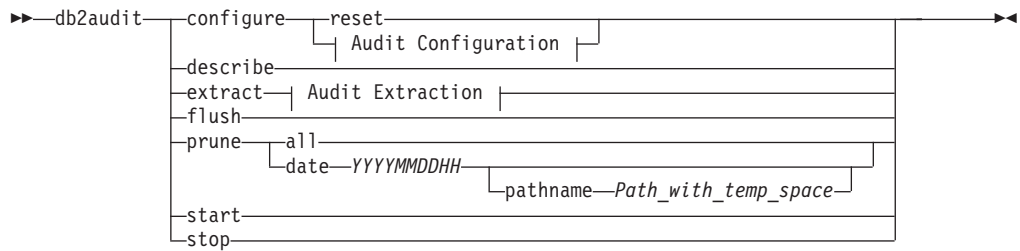
*sysadm*

### **Required Connection:**

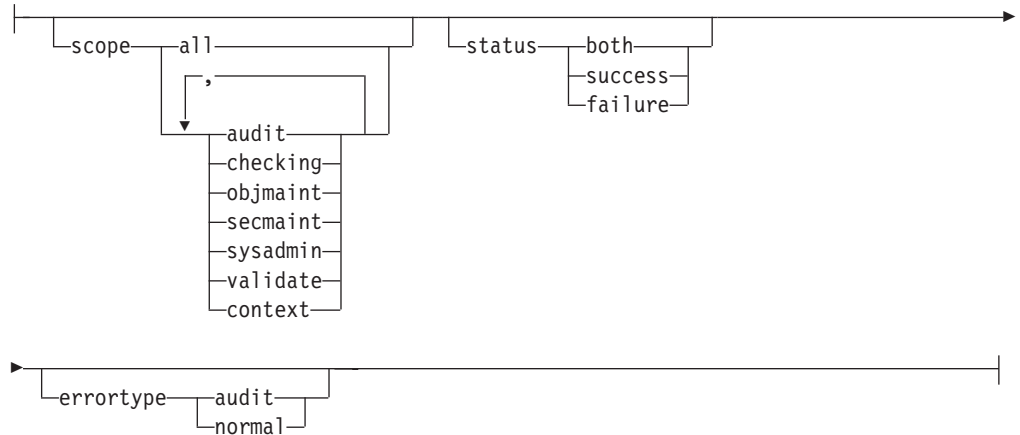
None.

### **Command syntax:**

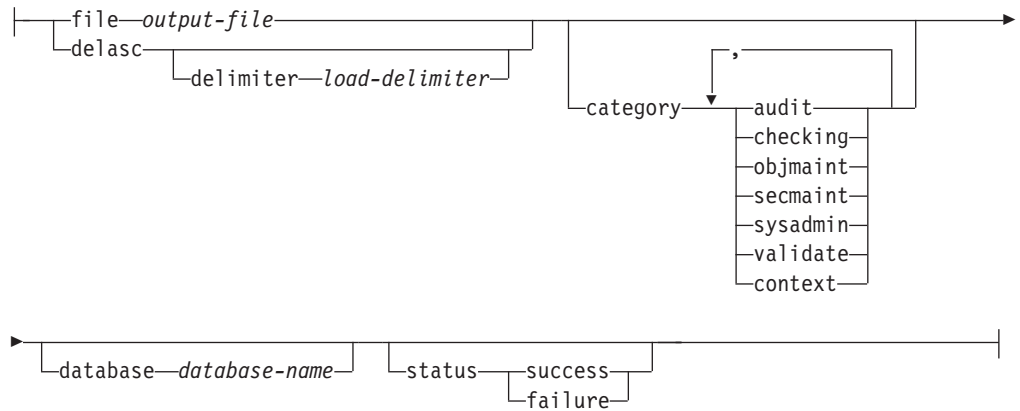
## db2audit - Audit facility administrator tool



### Audit Configuration:



### Audit Extraction:



### Command parameters:

#### configure

This parameter allows the modification of the db2audit.cfg configuration file in the instance's security subdirectory. Updates to this file can occur even when the instance is shut down. Updates occurring when the instance is active dynamically affect the auditing being done by DB2 database across all database partitions. The configure action on the configuration file causes the creation of an audit record if the audit facility has been started and the *audit* category of auditable events is being audited.

The following are the possible actions on the configuration file:

- RESET. This action causes the configuration file to revert to the initial configuration (where SCOPE is all of the categories except CONTEXT,



STATUS is FAILURE, ERRORTYPE is NORMAL, and the audit facility is OFF). This action will create a new audit configuration file if the original has been lost or damaged.

- SCOPE. This action specifies which category or categories of events are to be audited. This action also allows a particular focus for auditing and reduces the growth of the log. It is recommended that the number and type of events being logged be limited as much as possible, otherwise the audit log will grow rapidly. The default SCOPE is all categories except CONTEXT and may result in records being generated rapidly. In conjunction with the mode (synchronous or asynchronous), the selection of the categories may result in a significant performance reduction and significantly increased disk requirements.
- STATUS. This action specifies whether only successful or failing events, or both successful and failing events, should be logged. Context events occur before the status of an operation is known. Therefore, such events are logged regardless of the value associated with this parameter.
- ERRORTYPE. This action specifies whether audit errors are returned to the user or are ignored. The value for this parameter can be:
  - AUDIT. All errors including errors occurring within the audit facility are managed by DB2 database and all negative SQLCODEs are reported back to the caller.
  - NORMAL. Any errors generated by db2audit are ignored and only the SQLCODEs for the errors associated with the operation being performed are returned to the application.

### describe

This parameter displays to standard output the current audit configuration information and status.

**extract** This parameter allows the movement of audit records from the audit log to an indicated destination. If no optional clauses are specified, all of the audit records are extracted and placed in a flat report file. If *output\_file* already exists, an error message is returned.

The following are the possible options that can be used when extracting:

- FILE. The extracted audit records are placed in a file (*output\_file*). If no file name is specified, records are written to the `db2audit.out` file in the `security` subdirectory of `sql1ib`. If no directory is specified, *output\_file* is written to the current working directory.
- DELASC. The extracted audit records are placed in a delimited ASCII format suitable for loading into DB2 database relational tables. The output is placed in separate files: one for each category. The filenames are:
  - `audit.del`
  - `checking.del`
  - `objmaint.del`
  - `secmaint.del`
  - `sysadmin.del`
  - `validate.del`
  - `context.del`

These files are always written to the `security` subdirectory of `sql1ib`.

The DELASC choice also allows you to override the default audit character string delimiter (“0xff”) when extracting from the audit log. You would use DELASC DELIMITER followed by the new delimiter that you wish to use in preparation for loading into a table that will hold the

## db2audit - Audit facility administrator tool

audit records. The new load delimiter can be either a single character (such as !) or a four-byte string representing a hexadecimal number (such as 0xff).

- **CATEGORY.** The audit records for the specified categories of audit events are to be extracted. If not specified, all categories are eligible for extraction.
- **DATABASE.** The audit records for a specified database are to be extracted. If not specified, all databases are eligible for extraction.
- **STATUS.** The audit records for the specified status are to be extracted. If not specified, all records are eligible for extraction.

**flush** This parameter forces any pending audit records to be written to the audit log. Also, the audit state is reset in the engine from “unable to log” to a state of “ready to log” if the audit facility is in an error state.

**prune** This parameter allows for the deletion of audit records from the audit log. If the audit facility is active and the “audit” category of events has been specified for auditing, then an audit record will be logged after the audit log is pruned.

The following are the possible options that can be used when pruning:

- **ALL.** All of the audit records in the audit log are to be deleted.
- **DATE yyyymmddhh.** The user can specify that all audit records that occurred on or before the date/time specified are to be deleted from the audit log. The user may optionally supply a  
pathname

which the audit facility will use as a temporary space when pruning the audit log. This temporary space allows for the pruning of the audit log when the disk it resides on is full and does not have enough space to allow for a pruning operation.

**start** This parameter causes the audit facility to begin auditing events based on the contents of the db2audit.cfg file. In a partitioned DB2 database instance, auditing will begin on all database partitions when this clause is specified. If the “audit” category of events has been specified for auditing, then an audit record will be logged when the audit facility is started.

**stop** This parameter causes the audit facility to stop auditing events. In a partitioned DB2 database instance, auditing will be stopped on all database partitions when this clause is specified. If the “audit” category of events has been specified for auditing, then an audit record will be logged when the audit facility is stopped.

### Usage Notes:

- The audit facility must be stopped and started explicitly. When starting, the audit facility uses existing audit configuration information. Since the audit facility is independent of the DB2 database server, it will remain active even if the instance is stopped. In fact, when the instance is stopped, an audit record may be generated in the audit log.
- Ensure that the audit facility has been turned on by issuing the db2audit start command before using the audit utilities.
- There are different categories of audit records that may be generated. In the description of the categories of events available for auditing (below), you should notice that following the name of each category is a one-word keyword used to identify the category type. The categories of events available for auditing are:

- Audit (AUDIT). Generates records when audit settings are changed or when the audit log is accessed.
  - Authorization Checking (CHECKING). Generates records during authorization checking of attempts to access or manipulate DB2 database objects or functions.
  - Object Maintenance (OBJMAINT). Generates records when creating or dropping data objects.
  - Security Maintenance (SECMAINT). Generates records when granting or revoking: object or database privileges, or DBADM authority. Records are also generated when the database manager security configuration parameters SYSADM\_GROUP, SYSCTRL\_GROUP, or SYSMAINT\_GROUP are modified.
  - System Administration (SYSADMIN). Generates records when operations requiring SYSADM, SYSMAINT, or SYSCTRL authority are performed.
  - User Validation (VALIDATE). Generates records when authenticating users or retrieving system security information.
  - Operation Context (CONTEXT). Generates records to show the operation context when a database operation is performed. This category allows for better interpretation of the audit log file. When used with the log's event correlator field, a group of events can be associated back to a single database operation. For example, a query statement for dynamic queries, a package identifier for static queries, or an indicator of the type of operation being performed, such as CONNECT, can provide needed context when analyzing audit results. The SQL or XQuery statement providing the operation context might be very long and is completely shown within the CONTEXT record. This can make the CONTEXT record very large.
  - You can audit failures, successes, or both.
- Any operation on the database may generate several records. The actual number of records generated and moved to the audit log depends on the number of categories of events to be recorded as specified by the audit facility configuration. It also depends on whether successes, failures, or both, are audited. For this reason, it is important to be selective of the events to audit.

### Related tasks:

- "Creating DB2 audit data files" in *Administration Guide: Implementation*

### Related reference:

- "Audit facility usage" in *Administration Guide: Implementation*

## db2batch - Benchmark tool

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

This tool can work in both a single partition database and in a multiple partition database.

Through the tool's optional parameters you are able to control the number of rows to be fetched from the answer set, the number of fetched rows to be sent to the output file or standard output, and the level of performance information to be returned.

The output default is to use standard output. You can name the output file for the results summary.

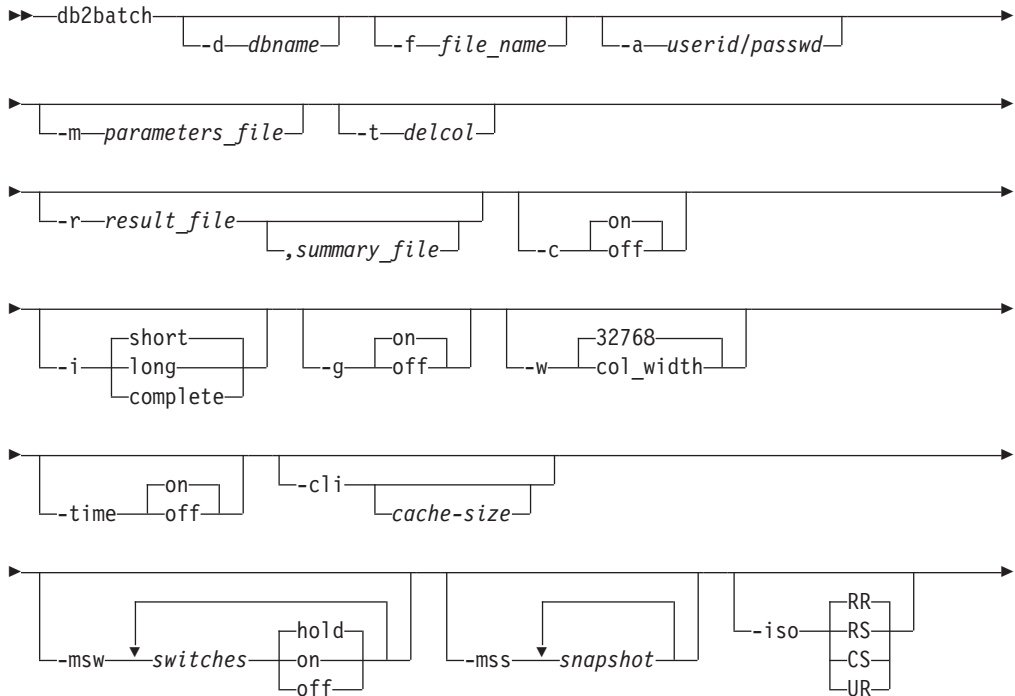
### Authorization:

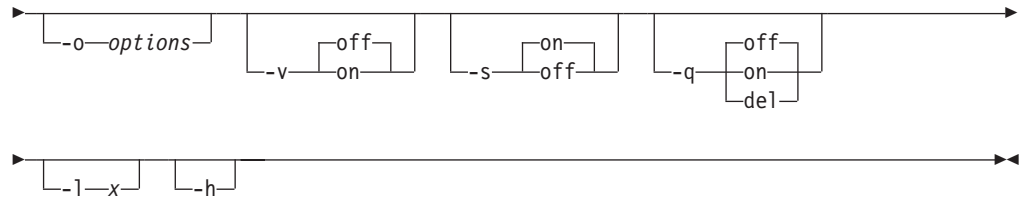
The same authority level as that required by the SQL statements or the XQuery statements to be read.

### Required connection:

None. This command establishes a database connection.

### Command syntax:



**Command parameters:****-d dbname**

An alias name for the database against which SQL statements and XQuery statements are to be applied. If this option is not specified, the value of the DB2DBDFT environment variable is used.

**-f file\_name**

Name of an input file containing SQL statements and XQuery statements. The default is standard input.

Identify comment text by adding two hyphens in front of the comment text, that is, -- <comment>. All text following the two hyphens until the end of the line is treated as a comment. Strings delimited with single or double quotes may contain two adjacent hyphens, and are treated as string constants rather than comments. To include a comment in the output, mark it as follows: --#COMMENT <comment>.

A *block* is a group of SQL statements and XQuery statements that are treated as one. By default, information is collected for all of the statements in the block at once, rather than 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. Blocks of queries can be included in a repeating loop by specifying a repeat count when defining the block, as follows: --#BGBLK [repeat\_count]. Statements in the block will be prepared only on the first iteration of the loop.

You can use #PARAM directives or a parameter file to specify the parameter values for a given statement and a given iteration of a block. See the -m option below for details.

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** *perf\_detail*

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

- 0 Do not return any timing information or monitoring snapshots.
- 1 Return elapsed time only.
- 2 Return elapsed time and a snapshot for the application.

- 3 Return elapsed time, and a snapshot for the database manager, the database, and the application.
- 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). The snapshot will not include hash join information.
- 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 buffer pools, table spaces and FCM (an FCM snapshot is only available in a multi-database-partition environment). The snapshot will not include hash join information.

The default value is 1. A value >1 is only valid on DB2 Version 2 and DB2 database servers, and is not currently supported on host machines.

### **ERROR\_STOP**

Specifies whether or not **db2batch** should stop running when a non-critical error occurs. Valid values are:

- no** Continue running when a non-critical error occurs. This is the default option.
- yes** Stop running when a non-critical error occurs.

### **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.

### **SNAPSHOT** *snapshot*

Specifies the monitoring snapshots to take. See the `-mss` option for the snapshots that can be taken.

### **TIMESTAMP**

Generates a time stamp.

### **TIMING**

Print timing information. Valid values are:

- ON** Timing information is printed. This is the default.
- OFF** Timing information is not printed.

### **-a** *userid/passwd*

Specifies the user ID and password used to connect to the database. The slash (/) must be included.

### **-m** *parameters\_file*

Specifies an input file with parameter values to bind to the SQL statement parameter markers before executing a statement. The default is to not bind parameters.

If a parameters file is used, then each line specifies the parameter values for a given statement and a given iteration of a block. If instead #PARAM directives are used, multiple values and even parameter ranges are specified in advance for each parameter of each statement, and on each iteration of the block a random value is chosen from the specified sets for each parameter. #PARAM directives and a parameters file cannot be mixed.

Parameter Value Format:

```
-36.6 'DB2' X'0AB2' G'...' NULL
12 'batch' x'32ef' N'...' null
+1.345E-6 'db2 batch' X'afd4' g'...' Null
```

Each parameter is defined like a SQL constant, and is separated from other parameters by whitespace. Non-delimited text represents a number, plain delimited (') text represents a single byte character string, 'x' or 'X' prefixed text enclosed in single quotes (') represents a binary string encoded as pairs of hex digits, 'g', 'G', 'n', or 'N' prefixed text enclosed in single quotes (') represents a graphic string composed of double byte characters, and 'NULL' (case insensitive) represents a null value. To specify XML data, use delimited (') text, such as '<last>Brown</last>'.

Parameter Input File Format:

Line X lists the set of parameters to supply to the Xth SQL statement that is executed in the input file. If blocks of statements are not repeated, then this corresponds to the Xth SQL statement that is listed in the input file. A blank line represents no parameters for the corresponding SQL statement. The number of parameters and their types must agree with the number of parameters and the types expected by the SQL statement.

Parameter Directive Format:

```
--#PARAM [single | start:end | start:step:end] [...]
```

Each parameter directive specifies a set of parameter values from which one random value is selected for each execution of the query. Sets are composed of both single parameter values and parameter value ranges. Parameter value ranges are specified by placing a colon (':') between two valid parameter values, with whitespace being an optional separator. A third parameter value can be placed between the start and end values to be used as a step size which overrides the default. Each parameter range is the equivalent of specifying the single values of 'start', 'start+step', 'start+2\*step', ... 'start+n\*step' where  $n$  is chosen such that 'start+n\*step'  $\geq$  'end' but 'start+(n+1)\*step'  $>$  'end'. While parameter directives can be used to specify sets of values for any type of parameter (even NULL), ranges are only supported on numerical parameter values (integers and decimal numbers).

**-t delcol**

Specifies a single character column separator. Specify -t TAB for a tab column delimiter or -t SPACE for a space column delimiter. By default, a space is used when the -q on option is set, and a comma is used when the -q del option is set.

**-r result\_file**

An output file that will contain the query results. If the optional *summary\_file* is specified, it will contain the summary table. The default is standard output.

## db2batch - Benchmark Tool

- c** Automatically commit changes resulting from each statement. The default is ON.
- i** Specifies to measure elapsed time intervals. Valid values are:
  - short** Measure the elapsed time to run each statement. This is the default.
  - long** Measure the elapsed time to run each statement including overhead between statements.
  - complete** Measure the elapsed time to run each statement where the prepare, execute, and fetch times are reported separately.
- g** Specifies whether timing is reported by block or by statement. Valid values are:
  - on** A snapshot is taken for the entire block and only block timing is reported in the summary table. This is the default.
  - off** A snapshot is taken and summary table timing is reported for each statement executed in the block.
- w** Specifies the maximum column width of the result set, with an allowable range of 0 to 2G. Data is truncated to this width when displayed, unless the data cannot be truncated. You can increase this setting to eliminate the warning CLI0002W and get a more accurate fetch time. The default maximum width is 32768 columns.
- time** Specifies whether or not to report the timing information. Valid values are:
  - on** Timing is reported. This is the default.
  - off** Timing is not reported.
- cli** Embedded dynamic SQL mode, previously the default mode for the **db2batch**, command is no longer supported. This command only runs in CLI mode. The **-cli** option exists for backwards compatibility. Specifying it (including the optional *cache-size* argument) will not cause errors, but will be ignored internally.
- msw** *switch*  
Sets the state of each specified monitor switch. You can specify any of the following: *uow*, *statement*, *table*, *bufferpool*, *lock*, *sort*, and *timestamp*. The special switch 'all' sets all of the above switches. For each switch that you specify you must choose one of:
  - hold** The state of the switch is unchanged. This is the default.
  - on** The switch is turned on.
  - off** The switch is turned off.
- mss** *snapshot*  
Specifies the monitoring snapshots that should be taken after each statement or block is executed, depending on the **-g** option. More than one snapshot can be taken at a time, with the information from all snapshots combined into one large table before printing. The possible snapshots are: *applinfo\_all*, *dbase\_applinfo*, *dcx\_applinfo\_all*, *db2*, *dbase*, *dbase\_all*, *dcx\_dbase*, *dcx\_dbase\_all*, *dbase\_remote*, *dbase\_remote\_all*, *agent\_id*, *dbase\_appls*, *appl\_all*, *dcx\_appl\_all*, *dcx\_appl\_handle*, *dcx\_dbase\_appls*,



dbase\_appls\_remote, appl\_remote\_all, dbase\_tables, appl\_locks\_agent\_id, dbase\_locks, dbase\_tablespaces, bufferpools\_all, dbase\_bufferpools, and dynamic\_sql.

The special snapshot 'all' takes all of the above snapshots. Any snapshots involving an appl ID are not supported in favour of their agent ID (application handle) equivalents. By default, no monitoring snapshots are taken.

- iso** Specifies the isolation level, which determines how data is locked and isolated from other processes while the data is being accessed. By default, **db2batch** uses the RR isolation level.

The TxnIsolation configuration keyword in the db2cli.ini file does not affect **db2batch**. To run this command with an isolation level other than RR, the -iso parameter must be specified.

**RR** Repeatable read (ODBC Serializable). This is the default

**RS** Read stability (ODBC Repeatable Read)

**CS** Cursor stability (ODBC Read Committed)

**UR** Uncommitted read (ODBC Read Uncommitted)

**-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** Do not return any timing information or monitoring snapshots.
- 1** Return elapsed time only.
- 2** Return elapsed time and a snapshot for the application.
- 3** Return elapsed time, and a snapshot for the database manager, the database, and the application.
- 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 buffer pools, table spaces and FCM (an FCM snapshot is only available in a multi-database-partition environment).

## db2batch - Benchmark Tool

The default value is 1. A value >1 is only valid on DB2 Version 2 and DB2 database servers, and is not currently supported on host machines.

**o** *query\_optimization\_class*

Sets the query optimization class. Valid values are 0, 1, 2, 3, 5, 7, or 9. The default is -1 to use the current optimization class.

**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:

**no** Run query only (default).

**explain**

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

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

**s** *error\_stop*

Specifies whether or not **db2batch** should stop running when a non-critical error occurs. Valid values are:

**no** Continue running when a non-critical error occurs. This is the default option.

**yes** Stop running when a non-critical error occurs.

- 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 with arithmetic and geometric means, the rows fetched, and the rows output.
- q** Query output. Valid values are:
- off** Output the query results and all associated information. This is the default.
- on** Output only query results in non-delimited format.
- del** Output only query results in delimited format.
- l x** Specifies the termination character (delimiter). The delimiter can be 1 or 2 characters. The default is a semi-colon (;).
- h, -u, -?** Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

### Examples:

- The following is sample output from the command `db2batch -d crystal -f update.sql`

```
* Timestamp: Thu Feb 02 2006 10:06:13 EST

* SQL Statement Number 1:
create table demo (c1 bigint, c2 double, c3 varchar(8));
* Elapsed Time is: 0.101091 seconds
```

-----  
 \* SQL Statement Number 2:

insert into demo values (-9223372036854775808, -0.000000000000005, 'demo');

\* Elapsed Time is: 0.002926 seconds

-----  
 \* SQL Statement Number 3:

insert into demo values (9223372036854775807, 0.000000000000005, 'demodemo');

\* Elapsed Time is: 0.005676 seconds

-----  
 \* SQL Statement Number 4:

select \* from demo;

| C1                   | C2                     | C3       |
|----------------------|------------------------|----------|
| -9223372036854775808 | -5.00000000000000E-015 | demo     |
| 9223372036854775807  | +5.00000000000000E-015 | demodemo |

\* 2 row(s) fetched, 2 row(s) output.

\* Elapsed Time is: 0.001104 seconds

-----  
 \* SQL Statement Number 5:

drop table demo;

\* Elapsed Time is: 0.176135 seconds

\* Summary Table:

| Type      | Number | Repetitions | Total Time (s) | Min Time (s) | Max Time (s) |
|-----------|--------|-------------|----------------|--------------|--------------|
| Statement | 1      | 1           | 0.101091       | 0.101091     | 0.101091     |
| Statement | 2      | 1           | 0.002926       | 0.002926     | 0.002926     |
| Statement | 3      | 1           | 0.005676       | 0.005676     | 0.005676     |
| Statement | 4      | 1           | 0.001104       | 0.001104     | 0.001104     |
| Statement | 5      | 1           | 0.176135       | 0.176135     | 0.176135     |

Arithmetic Mean Geometric Mean Row(s) Fetched Row(s) Output

|          |          |   |   |
|----------|----------|---|---|
| 0.101091 | 0.101091 | 0 | 0 |
| 0.002926 | 0.002926 | 0 | 0 |
| 0.005676 | 0.005676 | 0 | 0 |
| 0.001104 | 0.001104 | 2 | 2 |
| 0.176135 | 0.176135 | 0 | 0 |

\* Total Entries: 5  
 \* Total Time: 0.286932 seconds  
 \* Minimum Time: 0.001104 seconds  
 \* Maximum Time: 0.176135 seconds  
 \* Arithmetic Mean Time: 0.057386 seconds  
 \* Geometric Mean Time: 0.012670 seconds

-----  
 \* Timestamp: Thu Feb 02 2006 10:06:13 EST

## db2batch - Benchmark Tool

### Usage notes:

- All SQL statements must be terminated by a delimiter (default ';') set by the `--SET DELIMITER` command. This delimiter can be 1 or 2 characters.
- SQL statement length is limited only by available memory and the interface used. Statements can break over multiple lines, but multiple statements are not allowed on a single line.
- Input file line length is limited only by available memory.
- `c` automatically issues `CONNECT` and `CONNECT RESET` statements.
- `PAUSE` and `SLEEP` are timed when *long* is specified for the `-i` timing option.
- Explain tables must be created before explain options can be used.
- All command line options and input file statements are case insensitive with respect to **db2batch**.
- **db2batch** supports the following datatypes: INTEGER, CHAR, VARCHAR, LONG VARCHAR, FLOAT, SMALLINT, BIGINT, DECIMAL, DATE, TIME, TIMESTAMP, CLOB, GRAPHIC, VARGRAPHIC, LONGVARGRAPHIC, DBCLOB, BLOB, and XML.
- `--SET PERF_DETAIL perf_detail` (or `-o p perf_detail`) provides a quick way to obtain monitoring output. If the performance detail level is > 1, all monitor switches are turned on internally by db2batch. If more precise control of monitoring output is needed, use the options `-msw` and `-mss` (or `--SET SNAPSHOT`).

### Related concepts:

- "About isolation levels" in *Administration Guide: Planning*

### Related reference:

- "db2sql92 - SQL92 compliant SQL statement processor " on page 249

## 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, might be helpful in problem determination related to an application's bind file.

### Authorization:

None

### Required connection:

None

### Command syntax:



### Command parameters:

- 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.

### filespec

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

### Related concepts:

- “Binding embedded SQL packages to a database” in *Developing Embedded SQL Applications*
- “Displaying the contents of a bind file using the db2bfd tool” in *Troubleshooting Guide*

---

### db2ca - Start the Configuration Assistant

Starts the Configuration Assistant. The Configuration Assistant is a graphical interface that is used to manage DB2 database configuration such as database manager configuration, DB2 registry, node directory, database directory and DCS directory.

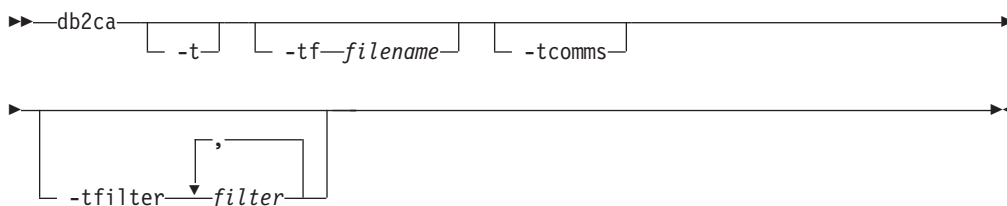
#### Authorization:

*sysadm*

#### Required Connection:

None.

#### Command syntax:



#### Command parameters:

- t** Turns on the GUI trace and sends the output to a console window. On Windows operating systems, the **db2ca** command does not have a console window. Therefore, this option has no effect on Windows operating systems.
- tf filename** Turns on the GUI trace and saves the output of the trace to the specified file. The output file is saved to <DB2 install path>\sql1lib\tools on Windows operating systems and to /home/<userid>/sql1lib/tools on Linux and UNIX-based systems.
- tcomms** Limits tracing to communications events.
- tfilter filter** Limits tracing to entries containing the specified filter or filters.

#### Related reference:

- "CATALOG DATABASE " on page 372
- "CATALOG DCS DATABASE " on page 375
- "CATALOG TCPIP/TCPIP4/TCPIP6 NODE " on page 387
- "db2set - DB2 profile registry " on page 245
- "GET DATABASE CONFIGURATION " on page 457
- "RESET DATABASE CONFIGURATION " on page 667
- "UPDATE DATABASE CONFIGURATION " on page 772

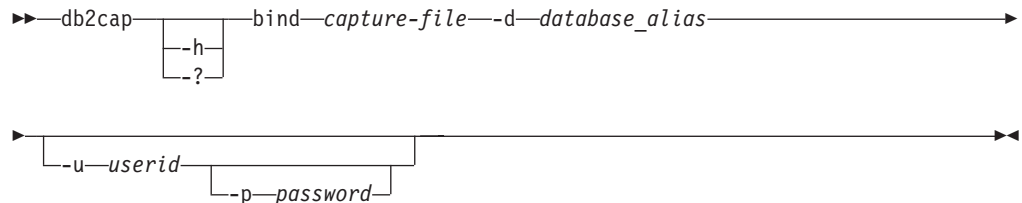
## db2cap - CLI/ODBC static package binding tool

Binds a capture file to generate one or more static packages. A capture file is generated during a static profiling session of a CLI/ODBC/JDBC application, and contains SQL statements that were captured during the application run. This utility processes the capture file so that it can be used by the CLI/ODBC/JDBC driver to execute static SQL for the application.

### Authorization:

- Access privileges to any database objects referenced by SQL statements recorded in the capture file.
- Sufficient authority to set bind options such as OWNER and QUALIFIER if they are different from the connect ID used to invoke the **db2cap** command.
- BINDADD authority if the package is being bound for the first time; otherwise, BIND authority is required.

### Command syntax:



### Command parameters:

**-h/-?** Displays help text for the command syntax.

#### **bind** *capture-file*

Binds the statements from the capture file and creates one or more packages.

#### **-d** *database\_alias*

Specifies the database alias for the database that will contain one or more packages.

#### **-u** *userid*

Specifies the user ID to be used to connect to the data source. If a user ID is not specified, a trusted authorization ID is obtained from the system.

#### **-p** *password*

Specifies the password to be used to connect to the data source.

### Usage notes:

This command must be entered in lowercase on UNIX platforms, but can be entered in either lowercase or uppercase on Windows operating systems.

This utility supports many user-specified bind options that can be found in the capture file. In order to change the bind options, open the capture file in a text editor.

The SQLERROR(CONTINUE) and the VALIDATE(RUN) bind options can be used to create a package.

## db2cap - CLI/ODBC Static Package Binding Tool

When using this utility to create a package, static profiling must be disabled.

The number of packages created depends on the isolation levels used for the SQL statements that are recorded in the capture file. The package name consists of up to a maximum of the first seven characters of the package keyword from the capture file, and one of the following single-character suffixes:

- 0 - Uncommitted Read (UR)
- 1 - Cursor Stability (CS)
- 2 - Read Stability (RS)
- 3 - Repeatable Read (RR)
- 4 - No Commit (NC)

To obtain specific information about packages, the user can:

- Query the appropriate SYSIBM catalog tables using the COLLECTION and PACKAGE keywords found in the capture file.
- View the capture file.

### **Related tasks:**

- “Creating static SQL with CLI/ODBC/JDBC Static Profiling” in *Call Level Interface Guide and Reference, Volume 1*



## db2cat - System catalog analysis

Analyzes the contents of packed descriptors. Given a database name and other qualifying information, this command will query the system catalogs for information and format the results. It must be issued on the server.

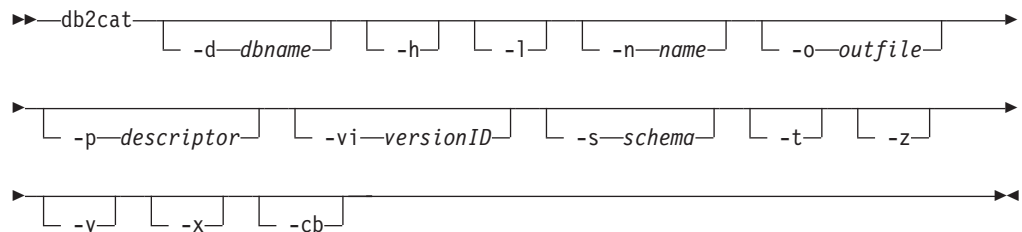
### Authorization:

None.

### Required Connection:

None.

### Command syntax:



### Command parameters:

#### **-d** *dbname*

*dbname* is the name of the database for which the command will query the system catalogs.

**-h** Displays usage information.

**-l** Turns on case sensitivity for the object name.

#### **-n** *name*

Specifies the name of the object.

#### **-o** *outfile*

Specifies the name of the output file.

#### **-p** *descriptor*

Specifies the name of the packed descriptor (pd) to display where *descriptor* is one of the following:

**check** Display table check constraints packed descriptor.

**rel** Display referential integrity constraint packed descriptor.

**table** Display table packed descriptor.

#### **summary**

Display summary table packed descriptor.

**trig** Display table trigger packed descriptor.

**view** Display view packed descriptor.

#### **remote**

Display remote non-relational data sources packed descriptor.

**ast** Display materialized query table packed descriptor.

## db2cat - System catalog analysis

### **routine**

Display routine packed descriptor.

### **sysplan**

Display package packed descriptor.

### **datatype**

Display structured type packed descriptor.

### **sequence**

Display sequence packed descriptor.

**esri** Display key transformation thread and index extension packed descriptor.

**event** Display event monitor packed descriptor.

**server** Display server packed descriptor.

**auth** Display privileges held by this grantee on this object.

### **-vi** *versionID*

Specifies the version ID of the package packed descriptor. **-vi** is only valid when **-p sysplan** is specified. If *versionID* is omitted, the default is the empty string.

### **-s** *schema*

Specifies the name of the object schema.

**-t** Displays terminal output.

**-z** Disables keystroke prompt.

**-v** Validates packed descriptor. This parameter is only valid for table packed descriptors.

**-x** Validates table space extentsize in catalogs (does not require a table name).

**-cb** Cleans orphan rows from SYSCAT.BUFFERPOOLNODES (does not require a table name).

### **Usage Notes:**

- Table name and table schema may be supplied in LIKE predicate form, which allows percent sign (%) and underscore (\_) to be used as pattern matching characters to select multiple sources with one invocation.
- Prompting will occur for all fields that are not supplied or are incompletely specified (except for the **-h** and **-l** options).
- If **-o** is specified without a file name, and **-t** is not specified, you will be prompted for a file name (the default name is `db2cat.out`).
- If neither **-o** nor **-t** is specified, you will be prompted for a file name (the default is terminal output).
- If **-o** and **-t** are both specified, the output will be directed to the terminal.

### **Related reference:**

- "System catalog views" in *SQL Reference, Volume 1*

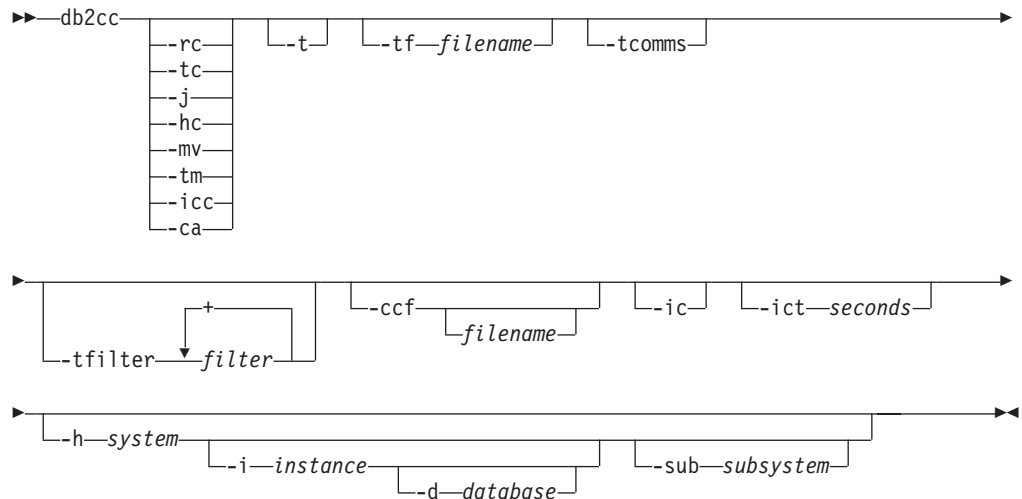
## db2cc - Start control center

Starts the Control Center. The Control Center is a graphical interface that is used to manage database objects (such as databases, tables, and packages) and their relationship to one another.

### Authorization:

*sysadm*

### Command syntax:



### Command parameters:

- rc** Opens the Replication Center.
- hc** Opens the Health Center.
- tc** Opens the Task Center.
- j** Opens the Journal.
- mv** Opens the Memory Visualizer.
- tm** Opens the Identify Indoubt Transaction Manager.
- icc** Opens the Information Catalog Manager.
- ca** Opens the Configuration Assistant.
- t** Turns on Control Center Trace for an initialization code. On Windows operating systems, the **db2cc** command does not have a console window. Therefore, this option has no effect on Windows operating systems.
- tf** Turns on Control Center Trace for an initialization code and saves the output of the trace to the specified file. The output file is saved to `<DB2 install path>\sqllib\tools` on Windows and to `/home/<userid>/sqllib/tools` on UNIX operating systems.
- tcomms**  
Limits tracing to communications events.
- tfilter filter**  
Limits tracing to entries containing the specified filter or filters.

## db2cc - Start Control Center

**-ccf file name**

Opens the Command Editor. If a filename is specified, the contents of this file are loaded into the Command Editor's Script page. When specifying a file name, you must provide the absolute path to the file.

**-ic** Opens the Information Center.

**-ict seconds**

Idle Connection Timer. Closes any idle connections in the pools maintained by the Control Center after the number of seconds specified. The default timer is 30 minutes.

**-h system**

Opens the Control Center in the context of a system.

**-i instance**

Opens the Control Center in the context of an instance.

**-d database**

Opens the Control Center in the context of a database.

**-sub subsystem**

Opens the Control Center in the context of a subsystem.

**Related reference:**

- "GET ADMIN CONFIGURATION " on page 441
- "RESET ADMIN CONFIGURATION " on page 663
- "UPDATE ADMIN CONFIGURATION " on page 756

## db2cfexp - Connectivity configuration export tool

Exports connectivity configuration information to an export profile, which can later be imported at another DB2 database workstation instance of similar instance type (i.e. client instance to client instance). The resulting profile will contain only configuration information associated with the current DB2 database instance. This profile can be referred to as a *client* configuration profile or a configuration profile of an *instance*.

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
- registry settings
- Common ODBC/CLI settings.

This utility is especially useful for exporting connectivity configuration information at workstations that do not have the DB2 Configuration Assistant installed, and in situations where multiple similar remote DB2 clients are to be installed, configured, and maintained (for example, cloning or making templates of client configurations).

### Authorization:

One of the following:

- *sysadm*
- *sysctrl*

### Command syntax:

```

▶▶ db2cfexp filename [TEMPLATE | 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 (i.e. client instance to client 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
- Common client settings in the database manager configuration
- Common client settings in the DB2 registry.

## db2cfexp - Connectivity Configuration Export Tool

### BACKUP

Creates a configuration profile of the DB2 database 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 DB2 registry
- All protocol information.

### MAINTAIN

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

### Related tasks:

- “Exporting and importing a profile” in *Installation and Configuration Supplement*

---

## 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 can contain connectivity items such as:

- Database information (including DB2 Connect and ODBC information)
- Node information
- Protocol information
- database manager configuration settings
- DB2 database registry settings
- Common ODBC/CLI settings.

This utility can be used to duplicate the connectivity information from another similar instance (i.e. client instance to client instance) that was configured previously. It is especially useful on workstations that do not have the DB2 Configuration Assistant (CA) installed, and in situations where multiple similar remote DB2 clients are to be installed, configured, and maintained (for example, cloning or making templates of client configurations). When cloning an instance, the profile imported should always be a client configuration profile that contains configuration information about one DB2 database instance only.

### 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 DB2 database or DB2 Connect product using the Configuration Assistant, Control Center, or **db2cfexp**.

### Related tasks:

- “Configuring database connections using a client profile with the Configuration Assistant” in *Quick Beginnings for DB2 Clients*
- “Exporting and importing a profile” in *Installation and Configuration Supplement*

---

### db2chglibpath - Modify the embedded runtime library search path

Modifies the embedded runtime library search path value within an executable or shared library file. It can be used to replace the embedded runtime library search path value with a new user-specified value when the existing value is no longer valid.

The **db2chglibpath** command can be used to replace the requirement for using operating system library search path environment variables such as LIBPATH (AIX), SHLIB\_PATH (HPPA, HPIPF) and LD\_LIBRARY\_PATH (AIX, SUN, HPPA64, HPIPF and Linux). This command is only supported on Linux and UNIX operating systems. It can be found under the DB2DIR/bin directory, where DB2DIR is the DB2 database installation location.

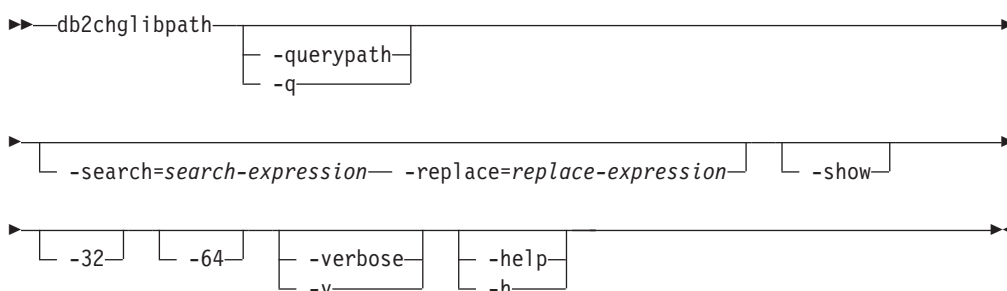
#### Prerequisites:

- Read and write access is required on the shared library or executable file to be modified.
- The binary has to have an embedded library path to start with, and the embedded path cannot be changed to anything bigger than the path already in the binary.
- The length of the user-specified value that is to replace the embedded runtime library search path value must not be greater than the existing value.
- This command directly modifies the binary code of the shared library or executable file and it is *strongly recommended* that you create a backup of the file before using the command.

#### Required Connection:

None.

#### Command syntax:



#### Command parameters:

##### **-querypath**

Specifies that a query should be performed without altering the embedded library path in the binary.

##### **-search=search-expression**

Specifies the expression to be searched for.

##### **-replace=replace-expression**

Specifies the expression that the *search-expression* is to be replaced with.

**-show** Specifies that the search and replace operations are to be performed without actually writing the changes to the file(s).



## db2chglibpath - Modify the embedded runtime library search path

**-32** Performs the operation if the binary type is 32-bit.

**-64** Performs the operation if the binary type is 64-bit.

**-verbose**

Displays information about the operations that are being performed.

**-help** Displays usage information.

**Examples:**

- To change the embedded runtime library search path value in the executable file named `myexecutable` from `/usr/opt/db2_08_01/lib` to `/u/usr1/sql1lib/lib32`, issue:

```
db2chglibpath -search=/usr/opt/db2_08_01/lib -replace=/u/usr1/sql1lib/lib32
/mypath/myexecutable
```

Note that the length of the new value is the same as that of the original value.

**Usage notes:**

- This command is only to be used for updating DB2 database application executables and DB2 external routine shared library files when other methods for migrating applications and routines cannot be used or are unsuccessful. See the related links for topics on application and routine migration.
- This command is not supported under DB2 service contract agreements. It is provided as-is and as such, IBM is not responsible for its unintended or malicious use.
- This command does not create a backup of the shared library or executable file before modifying it. It is *strongly recommended* that a backup copy of the file be made prior to issuing this command.

**Related tasks:**

- “Migrating 32-bit database applications to run on 64-bit instances” in *Migration Guide*
- “Migrating C, C++, and COBOL routines” in *Migration Guide*
- “Migrating embedded SQL and CLI applications” in *Migration Guide*

---

### db2chgpath - Change embedded runtime path

Used by DB2 installer on Linux and UNIX-based systems to update the embedded runtime path in the related DB2 library and executable files. The command can be reissued under the direction of IBM DB2 support if there were errors related to the command during the DB2 installation.

**Authorization:**

Root authority.

**Required Connection:**

None.

**Command syntax:**

```
▶▶ db2chgpath [-d] [-f file-name]
```

**Command parameters:**

- d** Turns debug mode on. Use this option only when instructed by DB2 Support.
- f file-name** Specifies a specific file name to update the runtime path. *file-name* should have the path name relative to the base of the current DB2 product install location.

**Examples:**

- To check all files under the DB2 product install path and do a runtime path update, issue:  
`<DB2 installation path>/install/db2chgpath`
- To update the path for a specific file called `libdb2.a` which is under `<DB2 installation path>/lib64` directory, issue:  
`<DB2 installation path>/install/db2chgpath -f lib64/libdb2.a`

**Related tasks:**

- “Installing a DB2 product manually” in *Installation and Configuration Supplement*
- “Manually installing payload files (Linux and UNIX)” in *Installation and Configuration Supplement*

## db2ckbkp - Check backup

This utility can be used to test the integrity of a backup image and to determine whether or not the image can be restored. It can also be used to display the metadata stored in the backup header.

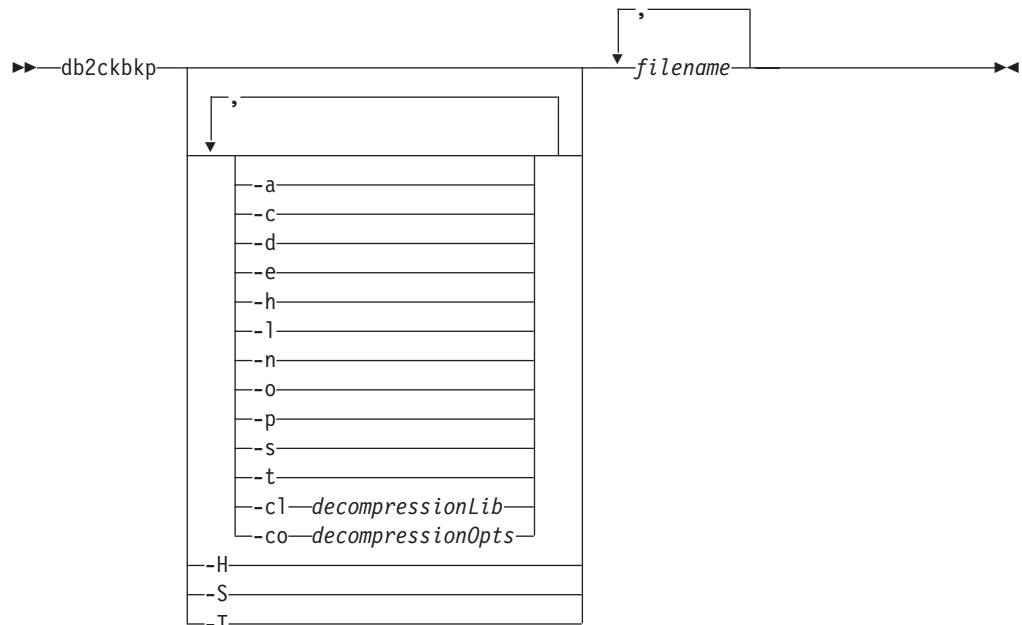
### Authorization:

Anyone can access the utility, but users must have read permissions on image backups in order to execute this utility against them.

### Required connection:

None

### Command syntax:



### Command parameters:

- a Displays all available information.
- c Displays results of checkbits and checksums.
- cl *decompressionLib*  
Indicates the name of the library to be used to perform the decompression. The name must be a fully qualified path referring to a file on the server. If this parameter is not specified, DB2 will attempt to use the library stored in the image. If the backup was not compressed, the value of this parameter will be ignored. If the specified library cannot be loaded, the operation will fail.
- co *decompressionOpts*  
Describes a block of binary data that will be passed to the initialization routine in the decompression library. DB2 will pass this string directly from the client to the server, so any issues of byte reversal or code page conversion will have to be handled by the decompression library. If the

## db2ckbkp - Check Backup

first character of the data block is '@', the remainder of the data will be interpreted by DB2 as the name of a file residing on the server. DB2 will then replace the contents of *string* with the contents of this file and will pass this new value to the initialization routine instead. The maximum length for string is 1024 bytes.

- d Displays information from the headers of DMS table space data pages.
- e Extracts pages from an image to a file. To extract pages, you will need an input and an output file. The default input file is called extractPage.in. You can override the default input file name by setting the DB2LISTFILE environment variable to a full path. The format of the input file is as follows:

For SMS table spaces:

```
S <tbspID> <objID> <objType> <startPage> <numPages>
```

**Notes:**

1. <startPage> is an object page number that is object-relative.

For DMS table spaces:

```
D <tbspID> <objType> <startPage> <numPages>
```

**Notes:**

1. <objType> is only needed if verifying DMS load copy images.
2. <startPage> is an object page number that is pool-relative.

For log files:

```
L <log num> <startPos> <numPages>
```

For other data (for example, initial data):

```
O <objType> <startPos> <numBytes>
```

The default output file is extractPage.out. You can override the default output file name by setting the DB2EXTRACTFILE environment variable to a full path.

- h Displays media header information including the name and path of the image expected by the restore utility.
- H Displays the same information as -h but only reads the 4K media header information from the beginning of the image. It does not validate the image. This option cannot be used in combination with any other options.
- l Displays log file header (LFH) and mirror log file header (MFH) data.
- n Prompt for tape mount. Assume one tape per device.
- o Displays detailed information from the object headers.
- p Displays the number of pages of each object type. This option will not show the number of pages for all different object types if the backup was done for DMS tablespaces data. It only shows the total of all pages as SQLUDMSTABLESPACEDATA. The object types for SQLUDMSLOBDATA and SQLUDMSLONGDATA will be zero for DMS tablespaces.
- s Displays the automatic storage paths in the image.
- S Displays the same information as -s but does not validate the image. This option cannot be used in combination with any other options.
- t Displays table space details, including container information, for the table spaces in the image.

**-T** Displays the same information as **-t** but does not validate the image. This option cannot be used in combination with any other options.

**filename**

The name of the backup image file. One or more files can be checked at a time.

**Notes:**

1. If the complete backup consists of multiple objects, the validation will only succeed if **db2ckbkp** is used to validate all of the objects at the same time.
2. When checking multiple parts of an image, the first backup image object (.001) must be specified first.

**Examples:**

Example 1 (on UNIX platforms)

```
db2ckbkp SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.001
SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.002
SAMPLE.0.krodger.NODE0000.CATN0000.19990817150714.003
```

```
[1] Buffers processed: ##
[2] Buffers processed: ##
[3] Buffers processed: ##
Image Verification Complete - successful.
```

Example 2

```
db2ckbkp -h SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001
```

```
=====
MEDIA HEADER REACHED:
=====
Server Database Name -- SAMPLE2
Server Database Alias -- SAMPLE2
Client Database Alias -- SAMPLE2
Timestamp -- 19990818122909
Database Partition Number -- 0
Instance -- krodger
Sequence Number -- 1
Release ID -- 900
Database Seed -- 65E0B395
DB Comment's Codepage (Volume) -- 0
DB Comment (Volume) --
DB Comment's Codepage (System) -- 0
DB Comment (System) --
Authentication Value -- 255
Backup Mode -- 0
Include Logs -- 0
Compression -- 0
Backup Type -- 0
Backup Gran. -- 0
Status Flags -- 11
System Cats inc -- 1
Catalog Database Partition No. -- 0
DB Codeset -- IS08859-1
DB Territory --
LogID -- 1074717952
LogPath -- /home/krodger/krodger/NODE0000/
 SQL00001/SQLLOGDIR
Backup Buffer Size -- 4194304
Number of Sessions -- 1
Platform -- 0
```

## db2ckbkp - Check Backup

The proper image file name would be:  
SAMPLE2.0.krodger.NODE0000.CATN0000.19990818122909.001

[1] Buffers processed: ####  
Image Verification Complete - successful.

### Usage notes:

1. If a backup image was created using multiple sessions, **db2ckbkp** can examine all of the files at the same time. Users are responsible for ensuring that the session with sequence number 001 is the first file specified.
2. This utility can also verify backup images that are stored on tape (except images that were created with a variable block size). This is done by preparing the tape as for a restore operation, and then invoking the utility, specifying the tape device name. For example, on UNIX based systems:

```
db2ckbkp -h /dev/rmt0
```

and on Windows:

```
db2ckbkp -d \\.\tape1
```

3. If the image is on a tape device, specify the tape device path. You will be prompted to ensure it is mounted, unless option '-n' is given. If there are multiple tapes, the first tape must be mounted on the first device path given. (That is the tape with sequence 001 in the header).

The default when a tape device is detected is to prompt the user to mount the tape. The user has the choice on the prompt. Here is the prompt and options: (where the device I specified is on device path /dev/rmt0)

```
Please mount the source media on device /dev/rmt0.
Continue(c), terminate only this device(d), or abort this tool(t)?
(c/d/t)
```

The user will be prompted for each device specified, and when the device reaches the end of tape.

### Related reference:

- “db2adutl - Managing DB2 objects within TSM” on page 15

## db2ckmig - Database pre-migration tool

Verifies that a database can be migrated.

### Scope:

This command only affects the database partition on which it is executed. In a partitioned database environment, run the command on each database partition.

### Authorization:

*sysadm*

### Required connection:

None

### Command syntax:

```

▶▶ db2ckmig database -l filename -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:

When an instance is migrated with the **db2imigr** command, **db2ckmig** is implicitly called as part of the migration. If you choose to run **db2ckmig** manually, it must be run for each database after the DB2 instance is installed, but before the instance is migrated. On Linux and UNIX-based systems, this utility is located in the DB2DIR/bin directory, where DB2DIR is the location where the DB2 copy is installed.

On Windows platforms, if you select the 'migrate' option during installation, instances are migrated and the installation will prompt you to run **db2ckmig**. A message box will warn you that if you have a local database on your system, you should run **db2ckmig** from the CD (it is located in db2\Windows\Utilities). Once you see the message box, you can either choose to ignore the message or quit the installation process. Run **db2ckmig** and then continue the installation if there are no errors, otherwise quit the installation, fix the error and install again. If you select the 'Install New' option instead, you will have to run **db2imigr** to migrate the instance which in turn will also run **db2ckmig**.

**db2ckmig** will not run against databases which are catalogued as remote databases.

## db2ckmig - Database Pre-migration Tool

To verify the state of a database:

1. Log on as the instance owner.
2. Issue the **db2ckmig** command.
3. Check the log file. When the **db2imigr** command runs the **db2ckmig** command, the log file specified is the `migration.log` file in the instance home directory for Linux and UNIX-based systems, and in the current directory for Windows operating systems. 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. When the tool reports an unrecognized `sqlcode/reason` code, refer to the new documentation for the release that you are moving to for details.

### Related tasks:

- “Verifying that your databases are ready for migration” in *Migration Guide*



## db2ckrst - Check incremental restore image sequence

Queries the database history and generates a list of timestamps for the backup images that are required for an incremental restore. A simplified restore syntax for a manual incremental restore is also generated.

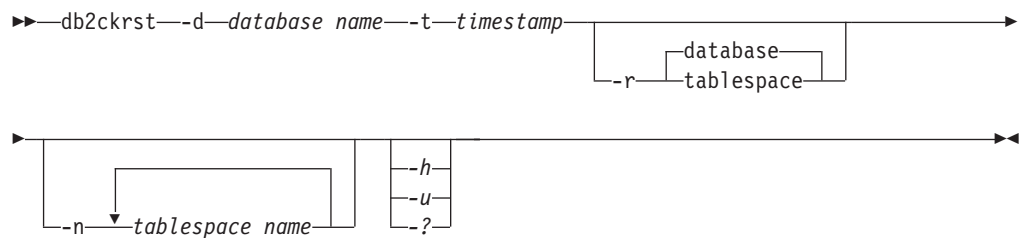
### Authorization:

None

### Required connection:

None

### Command syntax:



### Command parameters:

**-d** *database name*

Specifies the alias name for the database that will be restored.

**-t** *timestamp*

Specifies the timestamp for a backup image that will be incrementally restored.

**-r**

Specifies the type of restore that will be executed. The default is database. If TABLESPACE is chosen and no table space names are given, the utility looks into the history entry of the specified image and uses the table space names listed to do the restore.

**-n** *tablespace name*

Specifies the name of one or more table spaces that will be restored. If a database restore type is selected and a list of table space names is specified, the utility will continue as a table space restore using the table space names given.

**-h/-u/-?**

Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

### Examples:

```

db2ckrst -d mr -t 20001015193455 -r database
db2ckrst -d mr -t 20001015193455 -r tablespace
db2ckrst -d mr -t 20001015193455 -r tablespace -n tbsp1 tbsp2

```

```
> db2 backup db mr
```

```
Backup successful. The timestamp for this backup image is : 20001016001426
```

```
> db2 backup db mr incremental
```

## db2ckrst - Check Incremental Restore Image Sequence

Backup successful. The timestamp for this backup image is : 20001016001445

```
> db2ckrst -d mr -t 20001016001445
```

Suggested restore order of images using timestamp 20001016001445 for database mr.

```
=====
db2 restore db mr incremental taken at 20001016001445
db2 restore db mr incremental taken at 20001016001426
db2 restore db mr incremental taken at 20001016001445
=====
```

```
> db2ckrst -d mr -t 20001016001445 -r tablespace -n userspace1
Suggested restore order of images using timestamp 20001016001445 for
database mr.
```

```
=====
db2 restore db mr tablespace (USERSPACE1) incremental taken at
20001016001445
db2 restore db mr tablespace (USERSPACE1) incremental taken at
20001016001426
db2 restore db mr tablespace (USERSPACE1) incremental taken at
20001016001445
=====
```

### Usage notes:

The **db2ckrst** utility will not be enhanced for the rebuilding of a database. Due to the constraints of the history file, the utility will not be able to supply the correct list if several table spaces need to be restored from more than one image.

The database history must exist in order for this utility to be used. If the database history does not exist, specify the HISTORY FILE option in the RESTORE command before using this utility.

If the FORCE option of the PRUNE HISTORY command is used, you can delete entries that are required for automatic incremental restoration of databases. Manual restores will still work correctly. Use of this command can also prevent the dbckrst utility from being able to correctly analyse the complete chain of required backup images. The default operation of the PRUNE HISTORY command prevents required entries from being deleted. It is recommended that you do not use the FORCE option of the PRUNE HISTORY command.

This utility should not be used as a replacement for keeping records of your backups.

### Related tasks:

- "Restoring from incremental backup images" in *Data Recovery and High Availability Guide and Reference*

### Related reference:

- "PRUNE HISTORY/LOGFILE " on page 607
- "RESTORE DATABASE " on page 675

---

## 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.

**Related concepts:**

- “db2cli.ini initialization file” in *Call Level Interface Guide and Reference, Volume 1*

---

### 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 the *DB2 Command Window* icon.

This command is only available on Windows operating systems.

#### Authorization:

None

#### Required connection:

None

#### Command syntax:

►► db2cmd option-flag command ►►

#### Command parameters:

##### -c or /c

Execute command following the **-c** option in a new DB2 command window, and then terminate. For example, **db2cmd -c dir** causes the **dir** command to be invoked in a new DB2 command window, and then the DB2 command window closes.

##### -w or /w

Execute command following the **-w** option in a new DB2 command window, and wait for the new DB2 command window to be closed before terminating the process. For example, **db2cmd /w dir** invokes the **dir** command, and the process does not end until the new DB2 command window closes.

**-i or /i** Execute command following the **-i** option while sharing the same DB2 command window and inheriting file handles. For example, **db2cmd -i dir** executes the **dir** command in the same DB2 command window.

**-t or /t** Execute command following the **-t** option in a new DB2 CLP window with the specified command as the title of this new window.

#### Usage notes:

If DB21061E ("Command line environment not initialized.") is returned when bringing up the CLP-enabled DB2 window, 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
```

#### Related reference:

- "db2 - Command line processor invocation" on page 311

## db2dart - Database analysis and reporting tool

Examines databases for architectural correctness and reports any encountered errors.

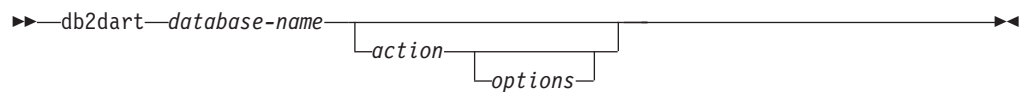
### Authorization:

*sysadm*

### Required connection:

None. **db2dart** must be run with no users connected to the database.

### Command syntax:



### Command parameters:

#### Inspection actions

- /DB** Inspects the entire database. This is the default option.
- /T** Inspects a single table. Requires two input values: a table space ID, and the table object ID or the table name.
- /TSF** Inspects only table space files and containers.
- /TSC** Inspects a table space's constructs, but not its tables. Requires one input value: table space ID.
- /TS** Inspects a single table space and its tables. Requires one input value: table space ID.
- /ATSC** Inspects constructs of all table spaces, but not their tables.

#### Data formatting actions

- /DD** Dumps formatted table data. Requires five input values: either a table object ID or table name, table space ID, page number to start with, number of pages, and verbose choice.
- /DI** Dumps formatted index data. Requires five input values: either a table object ID or table name, table space ID, page number to start with, number of pages, and verbose choice.
- /DM** Dumps formatted block map data. Requires five input values: either a table object ID or table name, table space ID, page number to start with, number of pages, and verbose choice.
- /DP** Dumps pages in hex format.
  - For permanent object in DMS tablespace, action /DP requires three input values consisting of tablespace ID, page number to start with, and number of pages.
  - For permanent object in SMS tablespace, action /DP requires five input values consisting of tablespace ID, object ID, page number to start with, number of pages, and object type.
- /DTSF** Dumps formatted table space file information.

## db2dart - Database Analysis and Reporting Tool

### **/DEMP**

Dumps formatted extent map page (EMP) information for a DMS table. Requires two input values: table space ID and the table object ID or table name.

### **/DDEL**

Dumps formatted table data in delimited ASCII format. Requires four input values: either a table object ID or table name, table space ID, page number to start with, and number of pages.

### **/DHWM**

Dumps high water mark information. Requires one input value: table space ID.

**/DXA** Dumps formatted XML column data in ASCII format. Requires five input values: either a table object ID or table name, table space ID, page number to start with, number of pages, and verbose choice.

**/DXH** Dumps formatted XML column data in HEX format. Requires five input values: either a table object ID or table name, table space ID, page number to start with, number of pages, and verbose choice.

### **/LHWM**

Suggests ways of lowering the high water mark. Requires two input values: table space ID and number of pages.

## **Repair actions**

**/ETS** Extends the table limit in a 4 KB table space (DMS only), if possible. Requires one input value: table space ID.

**/MI** Marks index as invalid. When specifying this parameter the database must be offline. Requires two input values: table space ID and index object ID

**/MT** Marks table with drop-pending state. When specifying this parameter the database must be offline. Requires three input values: table space ID, either table object ID or table name, and password.

**/IP** Initializes the data page of a table as empty. When specifying this parameter the database must be offline. Requires five input values: table name or table object ID, table space ID, page number to start with, number of pages, and password.

## **Change state actions**

### **/CHST**

Change the state of a database. When specifying this parameter the database must be offline. Requires one input value: database backup pending state.

## **Help**

**/H** Displays help information.

## **Input value options**

### **/OI** *object-id*

Specifies the object ID.

### **/TN** *table-name*

Specifies the table name.

**/TSI** *tablespace-id*

Specifies the table space ID.

**/ROW** *sum*

Identifies whether long field descriptors, LOB descriptors, and control information should be checked. You can specify just one option or add the values to specify more than one option.

**1** Checks control information in rows.

**2** Checks long field and LOB descriptors.

**/PW** *password*

Password required to execute the **db2dart** action. Contact DB2 Service for a valid password.

**/RPT** *path*

Optional path for the report output file.

**/RPTN** *file-name*

Optional name for the report output file.

**/PS** *number*

Specifies the page number to start with. The page number must be suffixed with p for pool relative. Specifying **/PS 0 /NP 0** will cause all pages in the specified object to be dumped.

**/NP** *number*

Specifies the number of pages. Specifying **/PS 0 /NP 0** will cause all pages in the specified object to be dumped.

**/V** *option*

Specifies whether or not the verbose option should be implemented. Valid values are:

**Y** Specifies that the verbose option should be implemented.

**N** Specifies that the verbose option should not be implemented.

**/SCR** *option*

Specifies type of screen output, if any. Valid values are:

**Y** Normal screen output is produced.

**M** Minimized screen output is produced.

**N** No screen output is produced.

**/RPTF** *option*

Specifies type of report file output, if any. Valid values are:

**Y** Normal report file output is produced.

**E** Only error information is produced to the report file.

**N** No report file output is produced.

**/ERR** *option*

Specifies type of log to produce in DART.INF, if any. Valid values are:

**Y** Produces normal log in DART.INF file.

**N** Minimizes output to log DART.INF file.

**E** Minimizes DART.INF file and screen output. Only error information is sent to the report file.

## db2dart - Database Analysis and Reporting Tool

### */WHAT DBBP option*

Specifies the database backup pending state. Valid values are:

**OFF** Off state.

**ON** On state.

### */QCK option*

Quick option. Only applies to /DB, /T, and /TS actions. Only inspects page 0 of the DAT objects and partially inspects the index objects (does not inspect BMP, LOB, LF objects and does not traverse the entirety of the DAT or INX objects).

### */TYP option*

Specifies the type of object. Valid values are:

**DAT** Object type is DAT.

**INX** Object type is INDEX.

**BKM** Object type is BMP.

### **Usage notes:**

1. When invoking the **db2dart** command, you can specify only one action. An action can support a varying number of options.
2. If you do not specify all the required input values when you invoke the **db2dart** command, you will be prompted for the values. For the /DDEL and /IP actions, the options cannot be specified from the command line, and must be entered when prompted by **db2dart**.
3. The /ROW, /RPT, /RPTN, /SCR, /RPTF, /ERR, and /WHAT DBBP options can all be invoked in addition to the action. They are not required by any of the actions.
4. The /DB, /T and /TS options inspect the specified objects, including associated XML storage objects. The /DB option includes all XML storage objects in the database, the /T option includes XML storage objects associated with the specified table, and the /TS option inspects all XML storage objects whose parent objects exist in the specified table space. As well, the /DEMP option will dump formatted EMP information including that for associated XML storage objects.
5. When **db2dart** is run against a single table space, all dependent objects for a parent table in that table space are checked, irrespective of the table space in which the dependent objects reside. However, extent map page (EMP) information is not captured for dependent objects that reside outside of the specified table space. EMP information is captured for dependent objects found in the specified table space even when the parent object resides in a table space other than the one specified.

### **Related reference:**

- “rah and db2\_all command descriptions” in *Administration Guide: Implementation*



---

## db2daslevel - Show DAS level

Shows the current level of the DAS on the system. Output from this command goes to the console by default.

**Authorization:**

None.

**Required Connection:**

None.

**Command Syntax:**

▶▶—db2daslevel—————▶▶

**Related reference:**

- “dasmigr - Migrate the DB2 administration server ” on page 6
- “dasupdt - Update DAS” on page 8
- “dasauto - Autostart DB2 administration server ” on page 3
- “dascrt - Create a DB2 administration server ” on page 4
- “dasdrop - Remove a DB2 administration server” on page 5

## 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.

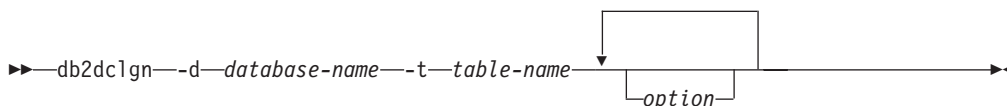
### Authorization:

None

### Required connection:

None

### Command syntax:



### 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 `sqlbchar` 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.

**-z encoding**

Specifies the encoding the coding convention in accordance to the

## db2dclgn - Declaration Generator

particular server. Encoding can be either LUW or OS390. If OS390 is specified, the generated file would look identical to a file generated by OS390.

### Related tasks:

- “Declaring Host Variables with the db2dclgn Declaration Generator” in *Developing Embedded SQL Applications*

## db2diag - db2diag.log analysis tool

Filters and formats the db2diag.log file.

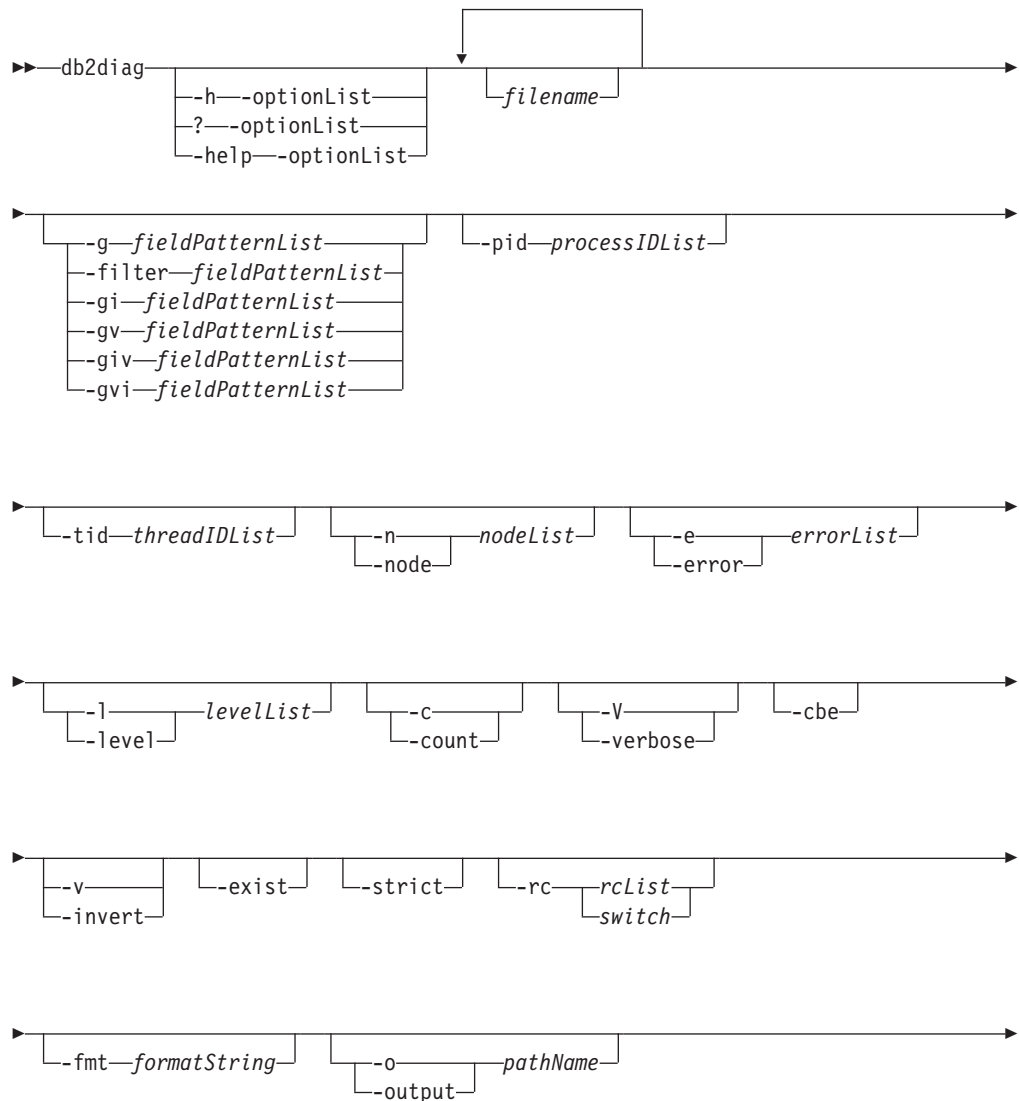
### Authorization:

None.

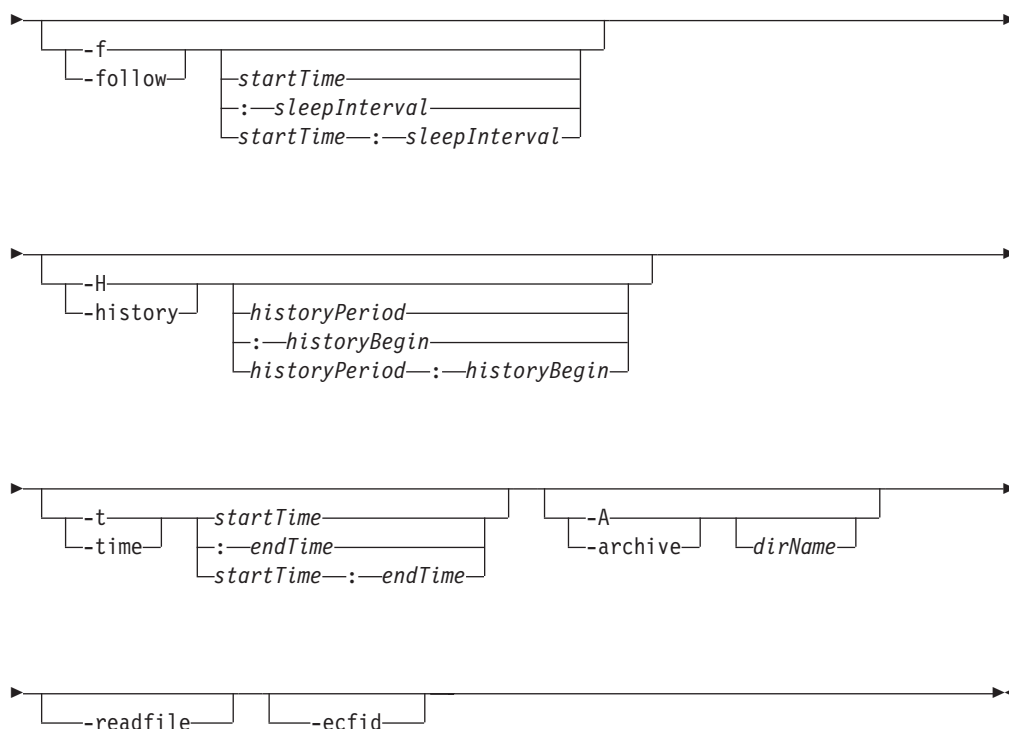
### Required connection:

None.

### Command syntax:



## db2diag - db2diag.log analysis tool



### Command parameters:

#### *filename*

Specifies one or more space-separated path names of DB2 diagnostic logs to be processed. If the file name is omitted, the db2diag.log file from the current directory is processed. If the file is not found, a directory set by the DIAGPATH variable is searched.

#### **-h/-help/?**

Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed. If a list of options, *optionList*, containing one or more comma separated command parameters is omitted, a list of all available options with short descriptions is displayed. For each option specified in the *optionList*, more detailed information and usage examples are displayed. Help output can be modified by using one of the following switches in place of the *optionList* argument to display more information about the tool and its usage:

**brief** Displays help information for all options without examples.

#### **examples**

Displays a few typical examples to assist in using the tool.

#### **tutorial**

Displays examples that describe advanced features.

**notes** Displays usage notes and restrictions.

**all** Displays complete information about all options, including usage examples for each option.

#### **-fmt** *formatString*

Formats the **db2diag** output using a format string, *formatString*, containing record fields in the form %field, %{field}, @field or @{field}. The %{field} and

@ {field} are used to separate a field name from the alphanumeric (or any other allowed character) that may follow the field name. All field names are case-insensitive. Field names can be shortened to the several first characters that are necessary to recognize a field name without ambiguity. In addition, aliases can be used for fields with long names. A prefix before a field name, %, or @, specifies whether a text preceding the field will be displayed (%) or not (@) if the field is empty.

The following fields are currently available (the same list is valid for the fields with the prefix @):

**%timestamp/%ts**

Time stamp. This field can be divided into its constituent fields: %tsyear,%tsmonth, %tsday, %tshour, %tsmin (minute),%tssec (second),%tsmsec (microsecond for UNIX operating systems, millisecond for Windows operating systems).

**%timezone/%tz**

Number of minutes difference from UTC (Universal Coordinated Time). For example, -300 is Eastern Time.

**%recordid/%recid**

Unique record ID.

**%audience**

Intended audience for a logged message. 'E' indicates external users (IBM customers, service analysts, and developers). 'I' indicates internal users (service analysts and developers). 'D' indicates debugging information for developers.

**%level**

Severity level of a message: Info, Warning, Error, Severe, or Event.

**%source**

Location from which the logged error originated: Origin, OS, Received, or Sent.

**%instance/%inst**

Instance name.

**%node**

Database partition server number.

**%database/%db**

Database name.

**%pid** Process ID.**%tid** Thread ID.**%process**

Name associated with the process ID, in double quotation marks. For example, "db2sysc.exe".

**%product**

Product name. For example, DB2 COMMON.

**%component**

Component name.

**%funcname**

Function name.

## db2diag - db2diag.log analysis tool

**%probe**  
Probe number.

**%function**  
Full function description: %prod, %comp, %funcname, probe:%probe.

**%appid**  
Application ID.

**%coordnode**  
Coordinator partition.

**%coordindex**  
Coordinator index.

**%apphdl**  
Application handle: %coordnode - %coordindex.

**%message/%msg**  
Error message.

**%calledprod**  
Product name of the function that returned an error.

**%calledcomp**  
Component name of the function that returned an error.

**%calledfunc**  
Name of the function that returned an error.

**%called**  
Full description of the function that returned an error: %calledprod, %calledcomp, %calledfunc.

**%rcval**  
Return code value (32 bytes).

**%rcdesc**  
Error description.

**%retcode/%rc**  
Return code returned by the function called: %rcval %rcdesc.

**%errno**  
System error number.

**%errname**  
System-specific error name.

**%oserror**  
Operating system error returned by a system call: %errno %errname.

**%callstack**  
Call stack.

**%datadesc**  
Data description.

**%dataobject**  
Data object.

**%data** Full data section of a message: %datadesc %dataobject.

**%argdesc**  
Argument description.



**%argobject**

Argument object.

**%arg** Arguments of a function call that returned an error: %argdesc  
%argobject.

**%startevent**

Start event description.

**%stopevent**

Stop event description.

**%changeevent**

Change event description.

To always display the text preceding a field name (for example, for the required fields), the % field prefix should be used. To display the text preceding a field name when this field contains some data, the@ prefix should be used. Any combination of required and optional fields with the corresponding text descriptions is allowed.

The following special characters are recognized within a format string: \n, \r, \f, \v, and \t.

In contrast to other fields, the data and argument fields can contain several sections. To output a specific section, add the [n] after the field name where n is a section number (1≤n≤64). For example, to output the first data object and the second data description sections, use %{dataobj}[1] and %{datadesc}[2]. When [n] is not used, all sections logged are output using pre-formatted logged data exactly as appears in a log message, so there is no need to add the applicable text description and separating newline before each data field, argument field, or section.

**-g** *fieldPatternList*

*fieldPatternList* is a comma-separated list of field-pattern pairs in the following format: fieldName operator searchPattern.

The operator can be one of the following:

- = Selects only those records that contain matches that form whole words. (Word search.)
- := Selects those records that contain matches in which a search pattern can be part of a larger expression.
- != Selects only non-matching lines. (Invert word match.)
- !:= Selects only non-matching lines in which the search pattern can be part of a larger expression.
- ^= Selects records for which the field value starts with the search pattern specified.
- !^= Selects records for which the field value does not start with the search pattern specified.

The same fields are available as described for the -fmt option, except that the% and @ prefixes are not used for this option

**-gi** *fieldPatternList*

Same as -g, but case-insensitive.

## db2diag - db2diag.log analysis tool

- gv** *fieldPatternList*  
Searches for messages that do not match the specified pattern.
- gvi/-giv** *fieldPatternList*  
Same as -gv, but case-insensitive.
- pid** *processIDList*  
Displays only log messages with the process IDs listed.
- tid** *threadIDList*  
Displays only log messages with the thread IDs listed.
- n/-node** *nodeList*  
Displays only log messages with the database partition numbers listed.
- e/-error** *errorList*  
Displays only log messages with the error numbers listed.
- l/-level** *levelList*  
Displays only log messages with the severity levels indicated.
- c/-count**  
Displays the number of records found.
- v/-invert**  
Inverts the pattern matching to select all records that do not match the specified pattern
- strict** Displays records using only one *field: value* pair per line. All empty fields are skipped.
- V/-verbose**  
Outputs all fields, including empty fields.
- exist** Defines how fields in a record are processed when a search is requested. If this option is specified, a field must exist in order to be processed.
- cbe** Common Base Event (CBE) Canonical Situation Data
- o/-output** *pathName*  
Saves the output to a file specified by a fully qualified *pathName*.
- f/follow**  
If the input file is a regular file, specifies that the tool will not terminate after the last record of the input file has been processed. Instead, it sleeps for a specified interval of time (*sleepInterval*), and then attempts to read and process further records from the input file as they become available.  
  
This option can be used when monitoring records being written to a file by another process. The *startTime* option, can be specified to show all the records logged after this time. The *startTime* option is specified using the following format: YYYY-MM-DD-hh.mm.ss.nnnnnn, where  
  
YYYY Specifies a year.  
MM Specifies a month of a year (01 through 12).  
DD Specifies a day of a month (01 through 31).  
hh Specifies an hour of a day (00 through 23).  
mm Specifies a minute of an hour (00 through 59).  
ss Specifies a second of a minute (00 through 59).

*nnnnnn*

Specifies microseconds on UNIX operating systems, or milliseconds on Windows operating systems.

Some or all of the fields that follow the year field can be omitted. If they are omitted, the default values will be used. The default values are 1 for the month and day, and 0 for all other fields.

If an exact match for the record time stamp does not exist in the diagnostic log file, the time closest to the time stamp specified will be used.

The *sleepInterval* option specifies a sleep interval in seconds. If a smaller time unit is required, it can be specified as a floating point value. The default value is 2 seconds

**-H/-history**

Displays the history of logged messages for the specified time interval. This option can be specified with the following options:

*historyPeriod*

Specifies that logged messages are displayed starting from the most recent logged record, for the duration specified by *historyPeriod*. The *historyPeriod* option is specified using the following format: Number *timeUnit*, where Number is the number of time units and *timeUnit* indicates the type of time unit: M (month), d (day), h (hour), m (minute), and s (second). The default value for Number is 30, and for *timeUnit* is m.

*historyPeriod:historyBegin*

Specifies that logged messages are displayed starting from the time specified by *historyBegin*, for the duration specified by *historyPeriod*.

The format is YYYY-MM-DD-*hh.mm.ss.nnnnnn*, where:

YYYY Specifies a year.

MM Specifies a month of a year (01 through 12).

DD Specifies a day of a month (01 through 31).

hh Specifies an hour of a day (00 through 23).

mm Specifies a minute of an hour (00 through 59).

ss Specifies a second of a minute (00 through 59).

*nnnnnn*

Specifies microseconds (UNIX operating systems) or milliseconds (Windows operating systems).

**-t/-time**

Specifies a time stamp value. This option can be specified with one or both of the following options:

*startTime*

Displays all messages logged after *startTime*.

*:endTime*

Displays all messages logged before *endTime*.

To display messages logged between *startTime* and *endTime*, specify *-t startTime:endTime*.

## db2diag - db2diag.log analysis tool

The format is *YYYY-MM-DD-hh.mm.ss.nnnnnn*, where:

- YYYY* Specifies a year.
- MM* Specifies a month of a year (01 through 12).
- DD* Specifies a day of a month (01 through 31).
- hh* Specifies an hour of a day (00 through 23).
- mm* Specifies a minute of an hour (00 through 59).
- ss* Specifies a second of a minute (00 through 59).
- nnnnnn* Specifies microseconds (UNIX operating systems) or milliseconds (Windows operating systems).

Some or all of the fields that follow the year field can be omitted. If they are omitted, the default values will be used. The default values are 1 for the month and day, and 0 for all other fields.

If an exact match for the record time stamp does not exist in the diagnostic log file, the time closest to the time stamp specified will be used.

### **-A/-archive** *dirName*

Archives a diagnostic log file. When this option is specified, all other options are ignored. If one or more file names are specified, each file is processed individually. A timestamp, in the format *YYYY-MM-DD-hh.mm.ss*, is appended to the file name.

You can specify the name of the file and directory where it is to be archived. If the directory is not specified, the file is archived in the directory where the file is located and the directory name is extracted from the file name.

If you specify a directory but no file name, the current directory is searched for the *db2diag.log* file. If found, the file will be archived in the specified directory. If the file is not found, the directory specified by the *DIAGPATH* configuration parameter is searched for the *db2diag.log* file. If found, it is archived in the directory specified.

If you do not specify a file or a directory, the current directory is searched for the *db2diag.log* file. If found, it is archived in the current directory. If the file is not found, the directory specified by the *DIAGPATH* configuration parameter is searched for the *db2diag.log* file. If found, it is archived in the directory specified by the *DIAGPATH* configuration parameter.

### **-readfile**

Forces reading from a diagnostic log file ignoring any terminal input. This option can be used in scripts to guarantee that **db2diag** will read from a file and not from a terminal especially in situations when *stdin* is disabled or when automated tools are used.

### **-rc** *rcList/switch*

Displays descriptions of DB2 internal error return codes for a space separated list, *rcList*, of the particular ZRC or ECF hexadecimal or negative decimal return codes. A full list of ZRC or ECF return codes can be displayed by specifying one of the following switches:

- zrc** Displays short descriptions of DB2 ZRC return codes.

- ecf** Displays short descriptions of DB2 ECF return codes.
- html** Displays short descriptions of DB2 ZRC return codes in the HTML format.

When this option is specified, all other options are ignored and output is directed to a display.

- ecfid** Displays function information extracted from the numeric ecfId. When this option is specified, all other options are ignored.

**Examples:**

To display all severe error messages produced by the process with the process ID (PID) 52356 and on node 1, 2 or 3, enter:

```
db2diag -g level=Severe,pid=952356 -n 1,2,3
```

To display all messages containing database SAMPLE and instance aabrashk, enter:

```
db2diag -g db=SAMPLE,instance=aabrashk
```

To display all severe error messages containing the database field, enter:

```
db2diag -g db:= -gi level=severe
```

To display all error messages containing the DB2 ZRC return code 0x87040055, and the application ID G916625D.NA8C.068149162729, enter:

```
db2diag -g msg:=0x87040055 -l Error | db2diag -gi appid^=G916625D.NA
```

To display all messages not containing the LOADID data, enter:

```
db2diag -gv data:=LOADID
```

To display only logged records not containing the LOCAL pattern in the application ID field, enter:

```
db2diag -gi appid!:=local or db2diag -g appid!:=LOCAL
```

All records that don't match will be displayed. To output only messages that have the application ID field, enter:

```
db2diag -gvi appid:=local -exist
```

To display all messages logged after the one with timestamp 2003-03-03-12.16.26.230520 inclusively, enter:

```
db2diag -time 2003-03-03-12.16.26.230520
```

To display severe errors logged for the last three days, enter:

```
db2diag -gi "level=severe" -H 3d
```

To display all log messages not matching the pdLog pattern for the funcname field, enter:

```
db2diag -g 'funcname!=pdLog' or db2diag -gv 'funcn=pdLog'
```

To display all severe error messages containing component name starting from the "base sys, enter:

```
db2diag -l severe | db2diag -g "comp^=base sys"
```

## db2diag - db2diag.log analysis tool

To view the growth of the db2diag.log file, enter: `db2diag -f db2diag.log` This displays all records written to the db2diag.log file in the current directory. Records are displayed as they added to the file The display continues until you press Ctrl-C.

To write the context of the db2diag.log into the db2diag\_123.log file located in the /home/user/Logs directory, enter:

```
db2diag -o /home/user/Logs/db2diag_123.log
```

To call **db2diag** from a Perl script using default settings, enter:

```
system("db2diag -readfile");
```

This will force **db2diag** to process db2diag.log from a directory specified by the DIAGPATH environment variable.

To read the db2diag.log1 file from a specified directory ignoring any terminal input, enter:

```
system("db2diag -readfile /u/usr/sql1lib/db2dump/db2diag.log1");
```

To display function information corresponding to ecfid = 0x1C30000E, enter:

```
db2diag -ecfid 0x1C30000E
```

which is equivalent to,

```
db2diag -ecfid 472907790
```

This will display function name, component and product name.

### Usage notes:

1. Each option can appear only once. They can be specified in any order and can have optional parameters. Short options can not be included together. For example, use `-l -e` and not `-le`.
2. By default, **db2diag** looks for the db2diag.log file in the current directory. If the file is not found, the directory set by the DIAGPATH registry variable is searched next. If the db2diag.log file is not found, **db2diag** returns an error and exits.
3. Filtering and formatting options can be combined on a single command line to perform complex searches using pipes. The formatting options `-fmt`, `-strict`, `-cbe`, and `-verbose` should be used only after all filtering is done to ensure that only original logged messages with standard fields will be filtered, not those fields either defined or omitted by the user. It is not necessary to use `-` when using pipes.
4. When pipes are used and one or more files names are specified on the command line, the **db2diag** input is processed differently depending on whether the `-` has been specified or not. If the `-` is omitted, input is taken from the specified files. In contrast, when the `-` option is specified, file names (even if present on the command line) are ignored and input from a terminal is used. When a pipe is used and a file name is not specified, the **db2diag** input is processed exactly the same way with or without the `-` specified on the command line.
5. The `-exist` option overwrites the default **db2diag** behavior for invert match searches when all records that do not match a pattern are output independent of whether they contain the proper fields or not. When the `-exist` option is specified, only the records containing fields requested are processed and output.

6. If the `-fmt` (format) option is not specified, all messages (filtered or not) are output exactly as they are written in the diagnostic log file. Output record format can be changed by using the `-strict`, `-cbe`, and `-verbose` options.
7. The `-fmt` option overwrites the `-strict`, `-cbe` and `-verbose` options.
8. Some restrictions apply when the `-cbe` option is specified and the `db2diag.log` file has been transferred over a network from the original computer. The **db2diag** tool collects information about DB2 and the computer host name locally, meaning that the DB2 version and the source or reporter componentID location field for the local system can be different from the corresponding values that were used on the original computer.
9. It is recommended to specify the `-readfile` option when using **db2diag** in scripts. It will ensure reading from a file ignoring any terminal input.
10. Ordinarily, the exit status is 0 if matches were found, and 1 if no matches were found. The exit status is 2 if there are syntax errors in the input data and patterns, the input files are inaccessible, or other errors are found.

### Related concepts:

- “Analyzing `db2diag.log` files using `db2diag`” in *Troubleshooting Guide*

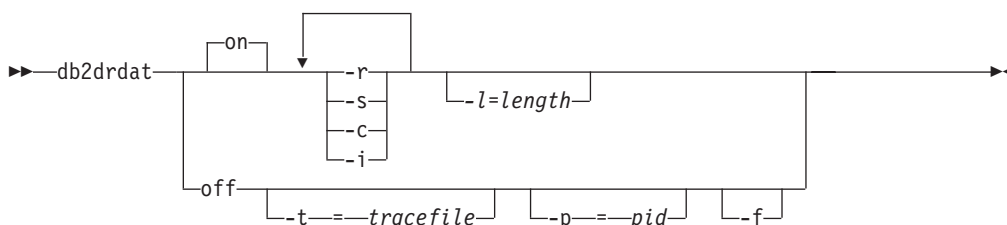
## db2drdat - DRDA trace

Allows the user to capture the DRDA data stream exchanged between a DRDA Application Requestor (AR) and the DB2 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. The *pid* to be traced can be found in the *agent* field returned by the LIST APPLICATIONS command.
- 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. If *tracefile* is not specified, messages are directed to db2drdat.dmp in the current directory.
- f** Formats communications buffers.

### Usage notes:

Do not issue **db2trc** commands while **db2drdat** is active.

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

1. **-r**
  - Type of DRDA request
  - Receive buffer



2. -s
  - Type of DRDA reply/object
  - Send buffer

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. 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.

**Related reference:**

- "LIST APPLICATIONS " on page 520

---

### db2drvmp - DB2 database drive map

Maps a database drive for Microsoft Cluster Server (MSCS). This command is available only on Windows platforms.

#### Authorization:

Read/write access to the Windows registry and the cluster registry.

#### Required connection:

Instance. The application creates a default instance attachment if one is not present.

#### Command syntax:

```
db2drvmp {add | drop | query | reconcile} dbpartition_number from_drive to_drive
```

#### Command parameters:

**add** Assigns a new database drive map.

**drop** Removes an existing database drive map.

**query** Queries a database map.

#### **reconcile**

Reapplies the database drive mapping to the registry when the registry contents are damaged or dropped accidentally.

#### *dbpartition\_number*

The database partition number. This parameter is required for add and drop operations. If this parameter is not specified for a reconcile operation, **db2drvmp** reconciles the mapping for all database partitions.

#### *from\_drive*

The drive letter from which to map. This parameter is required for add and drop operations. If this parameter is not specified for a reconcile operation, **db2drvmp** reconciles the mapping for all drives.

#### *to\_drive*

The drive letter to which to map. This parameter is required for add operations. It is not applicable to other operations.

#### Examples:

To set up database drive mapping from F: to E: for NODE0, issue the following command:

```
db2drvmp add 0 F E
```

To set up database drive mapping from E: to F: for NODE1, issue the following command:

```
db2drvmp add 1 E F
```

#### Usage notes:

1. Database drive mapping does not apply to table spaces, containers, or any other database storage objects.
2. Any setup of or change to the database drive mapping does not take effect immediately. To activate the database drive mapping, use the Microsoft Cluster Administrator tool to bring the DB2 resource offline, then online.
3. Using the TARGET\_DRVMAP\_DISK keyword in the DB2MSCS.CFG file will enable drive mapping to be done automatically.

**Related tasks:**

- “Migrating DB2 servers in Microsoft Cluster Server environments” in *Migration Guide*

### db2empfa - Enable multipage file allocation

Enables the use of multipage file allocation for a database. With multipage file allocation enabled for SMS table spaces, disk space is allocated one extent at a time rather than one page at a time.

#### Scope:

This command only affects the database partition 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 multipage file allocation is to be enabled.

#### Usage notes:

This utility:

- Connects to the database partition (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 in exclusive mode, it cannot be run concurrently on the catalog database partition, or on any other database partition.

Multipage file allocation can be enabled using **db2empfa** for databases that are created after the registry variable `DB2_NO_MPFA_FOR_NEW_DB` has been set.

#### Related concepts:

- “SMS table spaces” in *Administration Guide: Planning*

## 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 tables.

### Authorization:

The Event Analyzer reads data from event monitor tables stored with the database. For this reason, you must have the following authorization to access this data:

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

### Required connection:

Database connection

### Command syntax:

```

▶▶ db2eva [-db database-alias] [-evm evmon-name]

```

### Command parameters:

The **db2eva** parameters are optional. If you do not specify parameters, the Open Event Analyzer dialog box appears to prompt you for the database and event monitor name.

**-db** *database-alias*

Specifies the name of the database defined for the event monitor.

**-evm** *evmon-name*

Specifies the name of the event monitor whose traces are to be analyzed.

### Usage notes:

Without the required access, the user cannot retrieve any event monitor data.

There are two methods for retrieving event monitor traces:

1. The user can enter **db2eva** from the command line and the Open Event Analyzer Dialog box opens to let the user choose the database and event monitor names from the drop-down lists before clicking OK to open the Event Analyzer dialog box.
2. The user can specify the **-db** and **-evm** parameters from the command line and the Event Analyzer dialog opens on the specified database.

The Event Analyzer connects to the database, and issues a select target from SYSIBM.SYSEVENTTABLES to get the event monitor tables. The connection is then released after the required data has been retrieved.

The event analyzer can be used to analyze the data produced by an active event monitor. However, event monitor captured after the event analyzer has been invoked might not be shown. Turn off the event monitor before invoking the Event Analyzer to ensure data are properly displayed.

## db2eva - Event Analyzer

### Related reference:

- “Database system monitor interfaces” in *System Monitor Guide and Reference*

## db2evmon - Event monitor productivity tool

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

### Authorization:

None, unless connecting to the database (`-db -evm`); 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 instance is not already started when **db2evmon** is issued with the `-db` and `-evm` options, the command will start the instance.

If the instance is not already started when **db2evmon** is issued with the `-path` option, the command will not start the instance.

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.

### Related concepts:

## db2evmon - Event Monitor Productivity Tool

- “Event monitor self-describing data stream” in *System Monitor Guide and Reference*

### Related tasks:

- “Formatting file or pipe event monitor output from a command line” in *System Monitor Guide and Reference*



## db2evtbl - Generate event monitor target table definitions

Generates sample CREATE EVENT MONITOR SQL statements that can be used when defining event monitors that write to SQL tables.

### Authorization:

None.

### Required connection:

None.

### Command syntax:

```

▶▶ db2evtbl [-schema name] [-partitioned] -evm evmName

```

event type

The diagram shows the command syntax for db2evtbl. It starts with 'db2evtbl' followed by two optional options: '-schema name' and '-partitioned'. These are followed by '-evm evmName'. Below this, a separate line shows 'event type' with a bracket indicating it is a list of event types separated by commas.

### Command parameters:

#### -schema

Schema name. If not specified, the table names are unqualified.

#### -partitioned

If specified, elements that are only applicable for a partitioned database environment are also generated.

**-evm** The name of the event monitor.

#### event type

Any of the event types available on the CREATE EVENT MONITOR statement, for example, DATABASE, TABLES, TRANSACTIONS.

### Examples:

```
db2evtbl -schema smith -evm foo database, tables, tablespaces, bufferpools
```

### Usage notes:

Output is written to standard output.

Defining WRITE TO TABLE event monitors is more straightforward when using the **db2evtbl** tool. For example, the following steps can be followed to define and activate an event monitor.

1. Use **db2evtbl** to generate the CREATE EVENT MONITOR statement.
2. Edit the SQL statement, removing any unwanted columns.
3. Use the CLP to process the SQL statement. (When the CREATE EVENT MONITOR statement is executing, target tables are created.)
4. Issue SET EVENT MONITOR STATE to activate the new event monitor.

## db2evtbl - Generate Event Monitor Target Table Definitions

Since all events other than deadlock event monitors can be flushed, creating more than one record per event, users who do not use the FLUSH EVENT MONITOR statement can leave the element evmon\_flushes out of any target tables.

### Related concepts:

- “Event monitors” in *System Monitor Guide and Reference*

### Related reference:

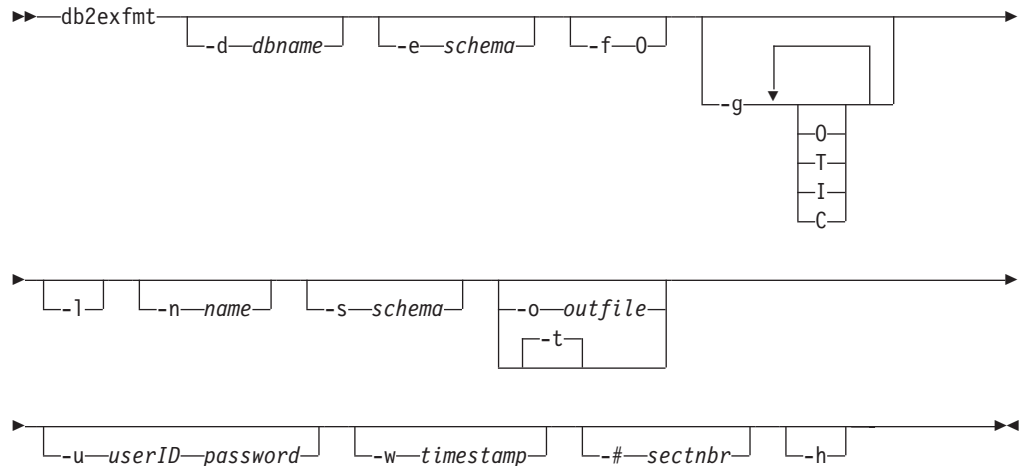
- “CREATE EVENT MONITOR statement” in *SQL Reference, Volume 2*
- “SET EVENT MONITOR STATE statement” in *SQL Reference, Volume 2*

## db2exfmt - Explain table format

You use the db2exfmt tool to format the contents of the explain tables. This tool is located in the misc subdirectory of the instance sql1lib directory.

To use the tool, you require read access to the explain tables being formatted.

### Command syntax:



### Command parameters:

- d *dbname***  
Name of the database containing packages.
- e *schema***  
Explain table SQL schema.
- f**      Formatting flags. In this release, the only supported value is 0 (operator summary).
- g**      Graph plan. If only -g is specified, a graph, followed by formatted information for all of the tables, is generated. Otherwise, any combination of the following valid values can be specified:
  - O**      Generate a graph only. Do not format the table contents.
  - T**      Include total cost under each operator in the graph.
  - I**      Include I/O cost under each operator in the graph.
  - C**      Include the expected output cardinality (number of tuples) of each operator in the graph.
- l**      Respect case when processing package names.
- n *name***  
Name of the source of the explain request (SOURCE\_NAME).
- s *schema***  
SQL schema or qualifier of the source of the explain request (SOURCE\_SCHEMA).
- o *outfile***  
Output file name.
- t**      Direct the output to the terminal.

## db2exfmt - Explain Table Format

- u** *userID password*  
When connecting to a database, use the provided user ID and password.  
Both the user ID and password must be valid according to naming conventions and be recognized by the database.
- w** *timestamp*  
Explain time stamp. Specify -1 to obtain the latest explain request.
- #** *sectnbr*  
Section number in the source. To request all sections, specify zero.
- h** Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.

### Usage notes:

You will be prompted for any parameter values that are not supplied, or that are incompletely specified, except in the case of the -h and the -l options.

If an explain table SQL schema is not provided, the value of the environment variable **USER** is used as the default. If this variable is not found, the user is prompted for an explain table SQL schema.

Source name, source SQL schema, and explain time stamp can be supplied in LIKE predicate form, which allows the percent sign (%) and the underscore (\_) to be used as pattern matching characters to select multiple sources with one invocation. For the latest explained statement, the explain time can be specified as -1.

If -o is specified without a file name, and -t is not specified, the user is prompted for a file name (the default name is db2exfmt.out). If neither -o nor -t is specified, the user is prompted for a file name (the default option is terminal output). If -o and -t are both specified, the output is directed to the terminal.

### Related concepts:

- “Explain tools” in *Performance Guide*
- “Guidelines for capturing explain information” in *Performance Guide*
- “Guidelines for using explain information” in *Performance Guide*

## db2exmig - Migrate explain tables

Migrates explain tables. The explain tables belonging to the user ID that is issuing the **db2exmig** command, or that is used to connect to the database, are migrated. The explain tables migration tool renames the existing explain tables, creates a new set of tables using the EXPLAIN.DDL, and copies the contents of the existing explain tables to the new tables. Finally, it drops the existing explain tables. The **db2exmig** command will preserve any user added columns on the explain tables.

### Authorization:

One of the following:

- *sysadm*
- *dbadm*

### Required Connection:

None.

### Command Syntax:

```
▶▶—db2exmig—-d—dbname—-e—explain_schema—┬──────────────────────────────────┴──▶▶
 └-u—userID password┘
```

### Command parameters:

- d *dbname*  
Specifies the database name.
- e *explain\_schema*  
Specifies the schema name of the explain tables to be migrated.
- u *userID password*  
Specifies the current user's ID and password.

### Related concepts:

- "The explain tables and organization of explain information" in *Performance Guide*

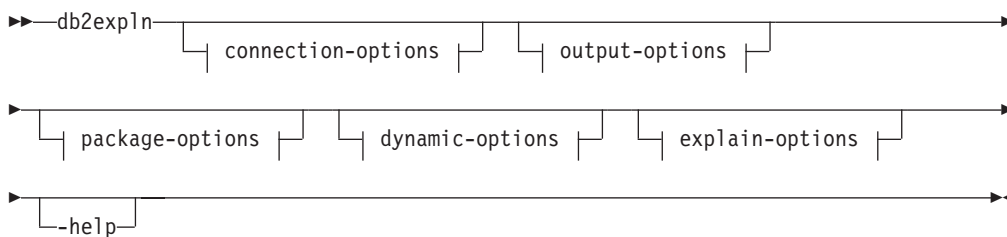
### Related tasks:

- "Migrating explain tables" in *Migration Guide*

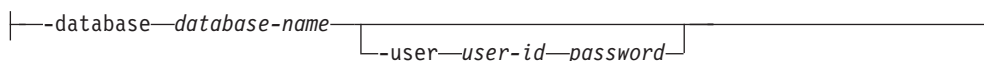
---

## db2expln - SQL and XQuery Explain

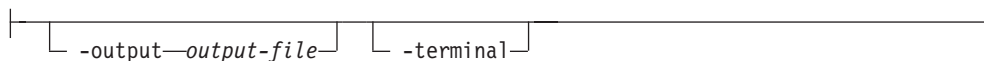
### Command syntax:



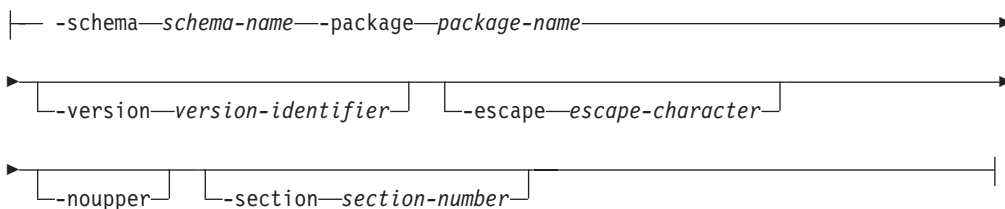
### connection-options:



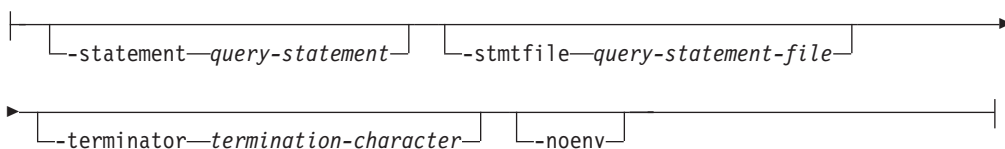
### output-options:



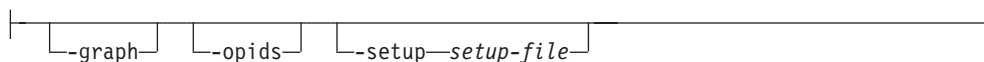
### package-options:



### dynamic-options:



### explain-options:



### Command parameters:

The options can be specified in any order.

### connection-options:

These options specify the database to connect to and any options necessary to make the connection. The connection options are required except when the **-help** option is specified.

**-database** *database-name*

The name of the database that contains the packages to be explained.

For backward compatibility, you can use **-d** instead of **-database**.

**-user** *user-id password*

The authorization ID and password to use when establishing the database connection. Both *user-id* and *password* must be valid according to DB2 naming conventions and must be recognized by the database.

For backward compatibility, you can use **-u** instead of **-user**.

### output-options:

These options specify where the db2expln output should be directed. Except when the **-help** option is specified, you must specify at least one output option. If you specify both options, output is sent to a file as well as to the terminal.

**-output** *output-file*

The output of db2expln is written to the file that you specify.

For backward compatibility, you can use **-o** instead of **-output**.

**-terminal**

The db2expln output is directed to the terminal.

For backward compatibility, you can use **-t** instead of **-terminal**.

### package-options:

These options specify one or more packages and sections to be explained. Only static queries in the packages and sections are explained.

As in a LIKE predicate, you can use the pattern matching characters, which are percent sign (%) and underscore (\_), to specify the *schema-name*, *package-name*, and *version-identifier*.

**-schema** *schema-name*

The SQL schema of the package or packages to be explained.

For backward compatibility, you can use **-c** instead of **-schema**.

**-package** *package-name*

The name of the package or packages to be explained.

For backward compatibility, you can use **-p** instead of **-package**.

**-version** *version-identifier*

The version identifier of the package or packages to be explained. The default version is the empty string.

**-escape** *escape-character*

The character, *escape-character* to be used as the escape character for pattern matching in the *schema-name*, *package-name*, and *version-identifier*.

For example, the db2expln command to explain the package TESTID.CALC% is as follows:

```
db2expln -schema TESTID -package CALC%
```

## db2expln - SQL and XQuery Explain

However, this command would also explain any other plans that start with CALC. To explain only the TESTID.CALC% package, you must use an escape character. If you specify the exclamation point (!) as the escape character, you can change the command to read: `db2expln -schema TESTID -escape ! -package CALC!% . . .`. Then the ! character is used as an escape character and thus !% is interpreted as the % character and not as the "match anything" pattern. There is no default escape character.

For backward compatibility, you can use **-e** instead of **-escape**.

To avoid problems, do not specify the operating system escape character as the **db2expln** escape character.

### **-noupper**

Specifies that the *schema-name*, *package-name*, and *version-identifier*, should not be converted to upper case before searching for matching packages.

By default, these variables are converted to upper case before searching for packages. This option indicates that these values should be used exactly as typed.

For backward compatibility, you can use **-l**, which is a lowercase L and not the number 1, instead of **-noupper**.

### **-section** *section-number*

The section number to explain within the selected package or packages.

To explain all the sections in each package, use the number zero (0). This is the default behavior. If you do not specify this option, or if *schema-name*, *package-name*, or *version-identifier* contain a pattern-matching character, all sections are displayed.

To find section numbers, query the system catalog view SYSCAT.STATEMENTS. Refer to the *SQL Reference* for a description of the system catalog views.

For backward compatibility, you can use **-s** instead of **-section**.

### **dynamic-options:**

These options specify one or more dynamic query statements to be explained.

### **-statement** *query-statement*

An SQL or XQuery query statement to be dynamically prepared and explained. To explain more than one statement, either use the **-stmtfile** option to provide a file containing the query statements to explain, or use the **-terminator** option to define a termination character that can be used to separate statements in the **-statement** option.

For compatibility with `dynexpln`, you can use **-q** instead of **-statement**.

### **-stmtfile** *query-statement-file*

A file that contains one or more query statements to be dynamically prepared and explained. By default, each line of the file is assumed to be a distinct query statement. If statements must span lines, use the **-terminator** option to specify the character that marks the end of an query statement.

For compatibility with `dynexpln`, you can use **-f** instead of **-stmtfile**.

### **-terminator** *termination-character*

The character that indicates the end of dynamic query statements. By default, the **-statement** option provides a single query statement and each



line of the file in the **-stmtfile** is treated as a separate query statement. The termination character that you specify can be used to provide multiple query statements with **-statement** or to have statements span lines in the **-stmtfile** file.

For compatibility with `dynexpln`, you can use **-z** instead of **-terminator**.

#### **-noenv**

Specifies that dynamic statements that alter the compilation environment should not be executed after they have been explained.

By default, `db2expln` will execute any of the following statements after they have been explained:

```
SET CURRENT DEFAULT TRANSFORM GROUP
SET CURRENT DEGREE
SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION
SET CURRENT QUERY OPTIMIZATION
SET CURRENT REFRESH AGE
SET PATH
SET SCHEMA
```

These statements make it possible to alter the plan chosen for subsequent dynamic query statements processed by `db2expln`.

If you specify **-noenv**, then these statements are explained, but not executed.

It is necessary to specify either **-statement** or **-stmtfile** to explain dynamic query. Both options can be specified in a single invocation of `db2expln`.

#### **explain-options:**

These options determine what additional information is provided in the explained plans.

**-graph** Show optimizer plan graphs. Each section is examined, and the original optimizer plan graph is constructed as presented by Visual Explain.

The generated graph may not match the Visual Explain graph exactly. It is possible for the optimizer graph to show some gaps, based on the information contained within the section plan.

For backward compatibility, you can specify **-g** instead of **-graph**.

**-opids** Display operator ID numbers in the explained plan.

The operator ID numbers allow the output from `db2expln` to be matched to the output from the explain facility. Not all operators have an ID number and that some ID numbers that appear in the explain facility output do not appear in the `db2expln` output.

For backward compatibility, you can specify **-i** instead of **-opids**.

**-help** Shows the help text for `db2expln`. If this option is specified no packages are explained.

Most of the command line is processed in the `db2exsrv` stored procedure. To get help on all the available options, it is necessary to provide **connection-options** along with **-help**. For example, use:

```
db2expln -help -database SAMPLE
```

For backward compatibility, you can specify **-h** or **-?**.

## db2expln - SQL and XQuery Explain

### **-setup** *setup-file*

A file that contains one or more statements needed to setup the environment for dynamic statements or for static statements that need to be recompiled (such as a static statement that references a declared global temporary table). Each statement in the file will be executed and any errors or warnings will be reported. The statements in the file are not explained.

### **Usage notes:**

Unless you specify the **-help** option, you must specify either package-options or dynamic-options. You can explain both packages and dynamic SQL with a single invocation of db2expln.

Some of the option flags above might have special meaning to your operating system and, as a result, might not be interpreted correctly in the db2expln command line. However, you might be able to enter these characters by preceding them with an operating system escape character. For more information, see your operating system documentation. Make sure that you do not inadvertently specify the operating system escape character as the db2expln escape character.

Help and initial status messages, produced by db2expln, are written to standard output. All prompts and other status messages produced by the explain tool are written to standard error. Explain text is written to standard output or to a file depending on the output option chosen.

### **Examples:**

To explain multiple plans with one invocation of db2expln, use the **-package**, **-schema**, and **-version** option and specify string constants for packages and creators with LIKE patterns. That is, the underscore (\_) can be used to represent a single character, and the percent sign (%) can be used to represent the occurrence of zero or more characters.

To explain all sections for all packages in a database named SAMPLE, with the results being written to the file **my.exp**, enter

```
db2expln -database SAMPLE -schema % -package % -output my.exp
```

As another example, suppose a user has a CLP script file called "statements.db2" and wants to explain the statements in the file. The file contains the following statements:

```
SET PATH=SYSIBM, SYSFUN, DEPT01, DEPT93@
SELECT EMPNO, TITLE(JOBID) FROM EMPLOYEE@
```

To explain these statements, enter the following command:

```
db2expln -database DEPTDATA -stmtfile statements.db2 -terminator @ -terminal
```

### **Related concepts:**

- "Description of db2expln and dynexpln output" in *Performance Guide*
- "Examples of db2expln and dynexpln output" in *Performance Guide*
- "SQL and XQuery Explain tools" in *Performance Guide*

## db2extsec - Set permissions for DB2 objects

Sets the permissions for DB2 objects (for example, files, directories, network shares, registry keys and services) on updated DB2 database system installations. In previous releases, this command was named **db2secv82**. The command name **db2secv82** is deprecated but can be used as an alternative name for **db2extsec**.

### Authorization:

- *sysadm*

### Required connection:

none

### Command syntax:

```

▶▶ db2extsec [/u—usergroup] [/a—adminingroup] [/r]

```

### Command parameters:

#### */u usergroup*

Specifies the name of the user group to be added. If this option is not specified, the default DB2 user group (DB2USERS) is used.

#### */a adminingroup*

Specifies the name of the administration group to be added. If this option is not specified, the default DB2 administration group (DB2ADMNS) is used.

#### */r*

Specifies that the changes made by previously running **db2extsec** should be reversed. If you specify this option, all other options are ignored. This option will only work if no other DB2 commands have been issued since the **db2extsec** command was issued.

### Related concepts:

- “Extended Windows security using DB2ADMNS and DB2USERS groups” in *Administration Guide: Implementation*

### Related tasks:

- “Adding your user ID to the DB2ADMNS and DB2USERS user groups (Windows)” in *Quick Beginnings for DB2 Servers*

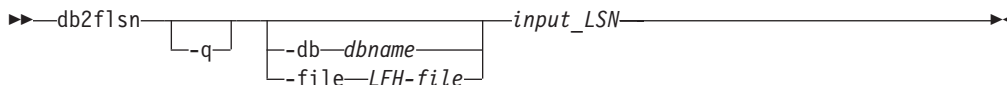
### db2flsn - Find log sequence number

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

#### Authorization:

None

#### Command syntax:



#### 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
- -106 Invalid database
- -500 Logical error

Other valid return codes are:

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

**-db *dbname***

Specifies the database name which you want to investigate.

**-file *LFH-name***

Specifies the full path of the LFH file including the file name.

**input\_LSN**

A 12 or 16 character string that represents the internal (6 or 8 byte) hexadecimal value with leading zeros.

#### Examples:

```
db2flsn 000000BF0030
Given LSN is contained in log page 2 in log file S0000002.LOG
```

```
db2flsn -q 000000BF0030
S0000002.LOG
```

```
db2flsn 000000BE0030
Given LSN is contained in log page 2 in log file S0000001.LOG
```

```
db2flsn -q 000000BE0030
S0000001.LOG
```

```
db2flsn -db flsntest 0000000000FA0000
```

## db2flsn - Find Log Sequence Number

Warning: the result is based on the last known log file size (6 4K pages starting from log extent 10). The input\_LSN might be before the database becomes recoverable.

Given LSN is contained in log page 2 in log file S0000002.LOG

```
db2flsn -q -db flsntest 0000000000FA0000
S0000002.LOG
```

```
db2flsn -file C:\DB2\NODE0000\SQL00001\SQLLOGCTL.LFH 0000000000FA4368
Given LSN is contained in log page 6 in log file S0000002.LOG
```

### Usage notes:

- If neither `-db` nor `-file` are specified, the tool assumes the LFH file is `SQLLOGCTL.LFH` in the current directory.
- 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 the `logarchmeth1` or `logarchmeth2` configuration parameters set to a value other than OFF.

### Related reference:

- “DB2 log records” in *Administrative API Reference*
- “SQLU\_LSN data structure” in *Administrative API Reference*

## db2fm - DB2 fault monitor

Controls the DB2 fault monitor daemon. You can use **db2fm** to configure the fault monitor.

This command is only available on UNIX operating systems.

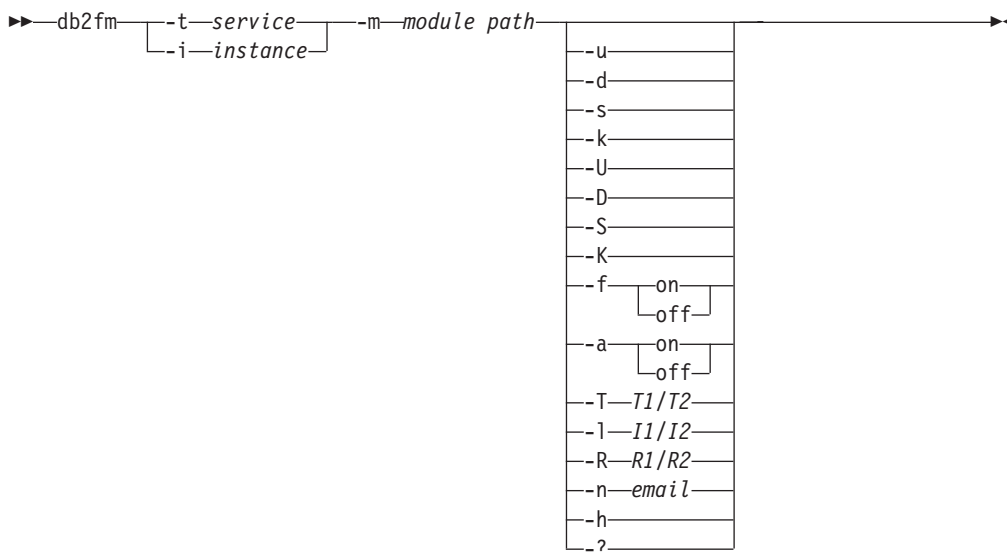
### Authorization:

Authorization over the instance against which you are running the command.

### Required connection:

None.

### Command syntax:



### Command parameters:

**-m** *module-path*

Defines the full path of the fault monitor shared library for the product being monitored. The default is \$INSTANCEHOME/sql/lib/libdb2gcf.

**-t** *service*

Gives the unique text descriptor for a service.

**-i** *instance*

Defines the instance of the service.

**-u** Brings the service up.

**-U** Brings the fault monitor daemon up.

**-d** Brings the instance down.

**-D** Brings the fault monitor daemon down.

**-k** Kills the service.

**-K** Kills the fault monitor daemon.

**-s** Returns the status of the service.

- S Returns the status of the fault monitor daemon. The status of the service or fault monitor can be one of the following
  - Not properly installed,
  - INSTALLED PROPERLY but NOT ALIVE,
  - ALIVE but NOT AVAILABLE (maintenance),
  - AVAILABLE, or
  - UNKNOWN
- f *on | off*  
Turns fault monitor on or off. If this option is set off, the fault monitor daemon will not be started, or the daemon will exit if it was running.
- a *on | off*  
Activates or deactivate fault monitoring. If this option if set off, the fault monitor will not be actively monitoring, which means if the service goes down it will not try to bring it back.
- T *T1/T2*  
Overwrites the start and stop time-out.  
For example:
  - -T 15/10 updates the two time-outs respectively
  - -T 15 updates the start time-out to 15 secs
  - -T /10 updates the stop time-out to 10 secs
- I *I1/I2*  
Sets the status interval and time-out respectively.
- R *R1/R2*  
Sets the number of retries for the status method and action before giving up.
- n *email*  
Sets the email address for notification of events.
- h Prints usage.
- ? Prints usage.

**Related concepts:**

- “Fault monitor facility for Linux and UNIX” in *Data Recovery and High Availability Guide and Reference*

---

### db2fs - First steps

Launches the First Steps interface which contains links to the functions users need to begin learning about and using DB2.

On UNIX operating systems, db2fs is located in the sqllib/bin directory. On Windows operating systems, db2fs.exe is located in the DB2PATH\bin directory.

One of the following browsers must be installed in order to issue the **db2fs** command:

- Internet Explorer 5.0 and up
- Mozilla 1.4 and up
- Firefox 1.0 and up
- Netscape 7.0 and up

#### Authorization:

*sysadm*

#### Command syntax:

##### For UNIX operating systems

```
▶▶ db2fs [-h] [-b browser]
```

##### For Windows operating systems

```
▶▶ db2fs
```

#### Command parameters:

##### For UNIX operating systems

**-h** Displays usage information.

**-b browser**

Specifies the browser to be used. If it is not specified, **db2fs** searches for a browser in the directories specified in PATH.

##### For Windows operating systems

None

#### Related concepts:

- “First Steps interface” in *Quick Beginnings for DB2 Servers*

#### Related tasks:

- “Verifying the installation of DB2 servers using First Steps (Linux and Windows)” in *Quick Beginnings for DB2 Servers*



## db2gcf - Control DB2 instance

Starts, stops, or monitors a DB2 instance, usually from an automated script, such as in an HA (high availability) cluster.

On UNIX operating systems, this command is located in `INSTHOME/sqllib/bin`, where `INSTHOME` is the home directory of the instance owner. On Windows systems, this command is located in the `sqllib\bin` subdirectory.

### Authorization:

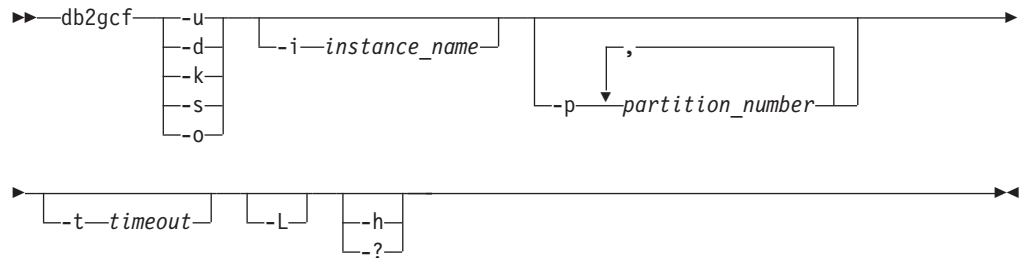
One of the following:

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

### Required connection:

None

### Command syntax:



### Command parameters:

- u** Starts specified database partition for specified instance on current database partition server (node).
- d** Stops specified database partition for specified instance.
- k** Removes all processes associated with the specified instance.
- s** Returns status of the specified database partition and the specified instance. The possible states are:
  - *Available*: The specified database partition for the specified instance is available for processing.
  - *Operable*: The instance is installed but not currently available.
  - *Not operable*: The instance will be unable to be brought to available state.
- o** Returns the default timeouts for each of the possible actions; you can override all these defaults by specifying a value for the `-t` parameter.
- i** *instance\_name*  
Instance name to perform action against. If no instance name is specified, the value of `DB2INSTANCE` is used. If no instance name is specified and `DB2INSTANCE` is not set, the following error is returned:  
db2gcf Error: Neither DB2INSTANCE is set nor instance passed.

## db2gcf - Control DB2 Instance

**-p** *partition\_number*

In a partitioned database environment, specifies database partition number to perform action against. If no value is specified, the default is 0. This value is ignored in a single-partition database environment.

**-t** *timeout*

Timeout in seconds. The **db2gcf** command will return unsuccessfully if processing does not complete within the specified period of time. There are default timeouts for each of the possible actions; you can override all these defaults by specifying a value for the **-t** parameter.

**-L** Enables error logging. Instance-specific information will be logged to `db2diag.log` in the instance log directory. Non-instance specific information will be logged to system log files.

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

### Examples:

1. The following example starts the instance *stevera* on partition 0:

```
db2gcf -u -p 0 -i stevera
```

The following output is returned:

```
Instance : stevera
DB2 Start : Success
Partition 0 : Success
```

2. The following example returns the status of the instance *stevera* on partition 0:

```
db2gcf -s -p 0 -i stevera
```

The following output is returned:

```
Instance : stevera
DB2 State
Partition 0 : Available
```

3. The following example stops the instance *stevera* on partition 0:

```
db2gcf -d -p 0 -i stevera
```

The following output is returned:

```
Instance : stevera
DB2 Stop : Success
Partition 0 : Success
```

### Usage notes:

When used together, the **-k** and **-p** parameters do not allow all processes to be removed from the specified partition. Rather, all processes on the instance (all partitions) will be removed.

### Related concepts:

- “High availability” in *Data Recovery and High Availability Guide and Reference*

## db2gov - DB2 governor

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

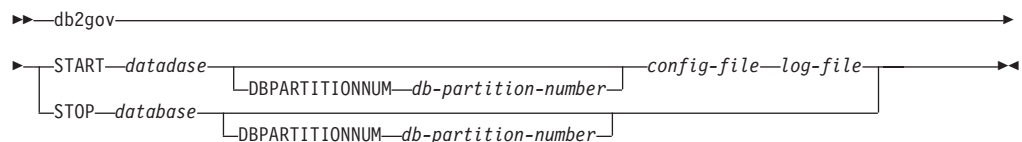
### Authorization:

One of the following:

- *sysadm*
- *sysctrl*

In an environment with an instance that has a `db2nodes.cfg` file defined, you might also require the authorization to invoke the **db2\_all** command. Environments with a `db2nodes.cfg` file defined include partitioned database environments as well as single-partition database environments that have a database partition defined in `db2nodes.cfg`.

### Command syntax:



### Command parameters:

#### START database

Starts the governor daemon to monitor the specified database. Either the database name or the database alias can be specified. The name specified must be the same as the one specified in the governor configuration file. One daemon runs for each database that is being monitored. In a partitioned database environment, one daemon runs for each database partition. If the governor is running for more than one database, there will be more than one daemon running at that database server.

#### DBPARTITIONNUM db-partition-number

Specifies the database partition on which to start or stop the governor daemon. The number specified must be the same as the one specified in the database partition configuration file.

#### config-file

Specifies the configuration file to use when monitoring the database. The default location for the configuration file is the `sql1ib` directory. If the specified file is not there, the front-end assumes that the specified name is the full name of the file.

#### log-file

Specifies the base name of the file to which the governor writes log records. The log file is stored in the `log` subdirectory of the `sql1ib` directory. The number of database partitions on which the governor is running is automatically appended to the log file name. For example, `mylog.0`, `mylog.1`, `mylog.2`.

#### STOP database

Stops the governor daemon that is monitoring the specified database. In a

## db2gov - DB2 Governor

partitioned database environment, the front-end utility stops the governor on all database partitions by reading the database partition configuration file `db2nodes.cfg`.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword `NODENUM` can be substituted for `DBPARTITIONNUM`.

### Related concepts:

- “The Governor utility” in *Performance Guide*

### Related tasks:

- “Configuring the Governor” in *Performance Guide*
- “Starting and stopping the governor” in *Performance Guide*

### Related reference:

- “db2govlg - DB2 governor log query ” on page 115

## db2govlg - DB2 governor log query

Extracts records of specified type from the governor log files. The DB2 governor monitors and changes the behavior of applications that run against a database.

### Authorization:

None

### Command syntax:

```

▶▶ db2govlg log-file dbpartitionnum db-partition-number
▶▶ rectype record-type

```

### Command parameters:

#### log-file

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

#### dbpartitionnum db-partition-number

Number of the database partition 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

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword nodenum can be substituted for dbpartitionnum.

### Related reference:

- “db2gov - DB2 governor ” on page 113

### db2gpmmap - Get distribution map

If a database is already set up and database partition groups defined for it, **db2gpmmap** gets the distribution map for the database table or the database partition group from the catalog partitioned database server.

#### Authorization:

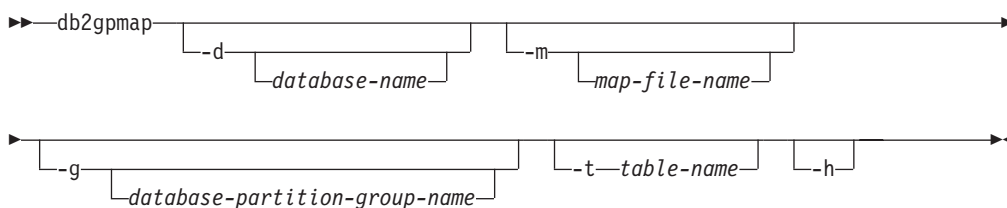
Both of the following:

- Read access to the system catalog tables
- BIND and EXECUTE package privileges on db2gpmmap.bnd

#### Required connection:

Before using **db2gpmmap** the database manager must be started and db2gpmmap.bnd must be bound to the database. If not already bound **db2gpmmap** will attempt to bind the file.

#### Command syntax:



#### Command parameters:

- d** Specifies the name of the database for which to generate a distribution map. If no database name is specified, the value of the DB2DBDFT environment variable is used. If DB2DBDFT is not set, the default is the SAMPLE database.
- m** Specifies the fully qualified file name where the distribution map will be saved. The default is db2split.map.
- g** Specifies the name of the database partition group for which to generate a distribution map. The default is IBMDEFAULTGROUP.
- t** Specifies the table name.
- h** Displays usage information.

#### Examples:

The following example extracts the distribution map for a table ZURBIE.SALES in database SAMPLE into a file called C:\pmaps\zurbie\_sales.map:

```
db2gpmmap -d SAMPLE -m C:\pmaps\zurbie_sales.map -t ZURBIE.SALES
```

#### Related concepts:

- “Distribution maps” in *Administration Guide: Planning*

## db2hc - Start health center

Starts the Health Center. The Health Center is a graphical interface that is used to view the overall health of database systems. Using the Health Center, you can view details and recommendations for alerts on health indicators and take the recommended actions to resolve the alerts.

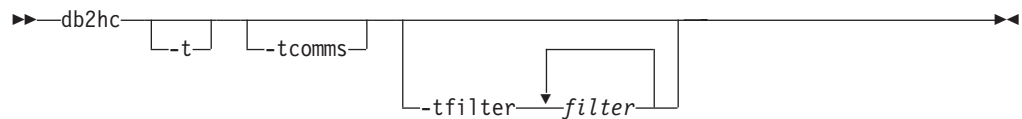
### Authorization:

No special authority is required for viewing the information. Appropriate authority is required for taking actions.

### Required Connection:

Instance

### Command Syntax:



### Command Parameters:

- t** Turns on NavTrace for initialization code. You should use this option only when instructed to do so by DB2 Support.
- tcomms** Limits tracing to communication events. You should use this option only when instructed to do so by DB2 Support.
- tfilter *filter*** Limits tracing to entries containing the specified filter or filters. You should use this option only when instructed to do so by DB2 Support.

### Related concepts:

- “Graphical tools for the health monitor” in *System Monitor Guide and Reference*

### Related tasks:

- “Configuring health indicators using the Health Center” in *System Monitor Guide and Reference*

### db2iauto - Auto-start instance

Enables or disables the auto-start of an instance after each system restart. This command is available on Linux and UNIX-based systems only.

**Authorization:**

One of the following:

- Root authority
- *sysadm*

**Required connection:**

None

**Command syntax:**

▶▶—db2iauto—-on  
-off—*instance-name*—▶▶

**Command parameters:**

- on** Enables auto-start for the specified instance.
- off** Disables auto-start for the specified instance.

*instance-name*

The login name of the instance.

**Related tasks:**

- “Auto-starting instances” in *Administration Guide: Implementation*



## db2iclus - Microsoft cluster server

Allows users to add, drop, migrate and unmigrate instances and DB2 administration servers (DAS) in a Microsoft Cluster Server (MSCS) environment. This command is only available on Windows platforms.

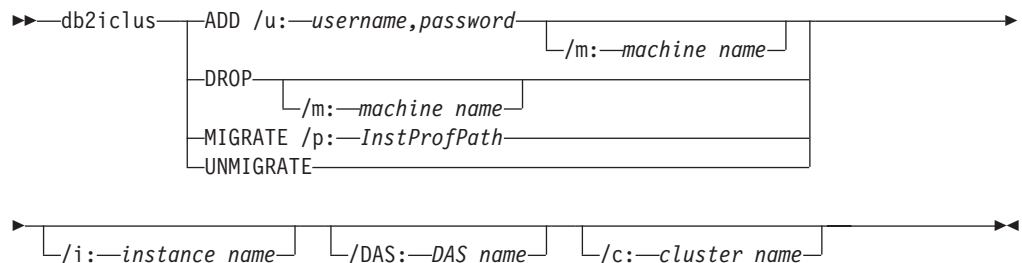
### Authorization:

Local administrator authority is required on the machine where the task will be performed. If adding a remote machine to an instance or removing a remote machine from an instance, local administrator authority is required on the target machine.

### Required connection:

None.

### Command syntax:



### Command parameters:

**ADD** Adds an MSCS node to a DB2 MSCS instance.

**DROP** Removes an MSCS node from a DB2 MSCS instance.

#### MIGRATE

Migrates a non-MSCS instance to an MSCS instance.

#### UNMIGRATE

Undoes the MSCS migration.

#### /DAS:DAS name

Specifies the DAS name. This option is required when performing the cluster operation against the DB2 administration server.

#### /c:cluster name

Specifies the MSCS cluster name if different from the default/current cluster.

#### /p:instance profile path

Specifies the instance profile path. This path must reside on a cluster disk so it is accessible when DB2 is active on any machine in the MSCS cluster. This option is required when migrating a non-MSCS instance to an MSCS instance.

#### /u:username,password

Specifies the account name and password for the DB2 service. This option is required when adding another MSCS node to the DB2 MSCS partitioned database instance.

## db2iclus - Microsoft Cluster Server

*/m:machine name*

Specifies the remote computer name for adding or removing an MSCS node.

*/i:instance name*

Specifies the instance name if different from the default/current instance.

### Examples:

This example shows the use of the **db2iclus** command to manually configure the DB2 instance to run in a hot standby configuration that consists of two machines, WA26 and WA27.

1. To start, MSCS and DB2 Enterprise Server Edition must be installed on both machines.
2. Create a new instance called DB2 on machine WA26:

```
db2icrt DB2
```

3. From the Windows Services dialog box, ensure that the instance is configured to start manually.
4. If the DB2 instance is running, stop it with the DB2STOP command.
5. Install the DB2 resource type from WA26:

```
c:>db2wolfi i
ok
```

If the **db2wolfi** command returns "Error : 183", then it is already installed. To confirm, the resource type can be dropped and added again. Also, the resource type will not show up in Cluster Administrator if it does not exist.

```
c:>db2wolfi u
ok
c:>db2wolfi i
ok
```

6. From WA26, use the **db2iclus** command to transform the DB2 instance into a clustered instance.

```
c:\>db2iclus migrate /i:db2 /c:mycluster /m:wa26 /p:p:\db2profs
```

```
DBI1912I The DB2 Cluster command was successful.
Explanation: The user request was successfully processed.
User Response: No action required.
```

The directory p:\db2profs should be on a clustered drive and must already exist. This drive should also be currently owned by machine WA26.

7. From WA26, use the **db2iclus** command to add other machines to the DB2 cluster list:

```
c:\>db2iclus add /i:db2 /c:mycluster /m:wa27
```

```
DBI1912I The DB2 Cluster command was successful.
Explanation: The user request was successfully processed.
User Response: No action required.
```

This command should be executed for each subsequent machine in the cluster.

8. From Cluster Administrator, create a new group called "DB2 Group".
9. From Cluster Administrator, move the Physical Disk resources Disk O and Disk P into DB2 Group.
10. From Cluster Administrator, create a new resource type of type "IP Address" called "mcs5" that resides on the Public Network. This resource should also belong to DB2 Group. This will be a highly available IP address, and this

address should not correspond to any machine on the network. Bring the IP Address resource type online and ensure that the address can be pinged from a remote machine.

11. From Cluster Administrator, create a new resource of type "DB2" that will belong to DB2 Group. The name of this resource must be exactly identical to the instance name, so it is called DB2 for this case. When Cluster Administrator prompts for dependencies associated with the DB2 resource, ensure it is dependent on Disk O, Disk P and mscs5.
12. Configure DB2 Group for fallback, if desired, via Cluster Administrator and using the DB2\_FALLBACK profile variable.
13. Create or restore all databases putting all data on Disk O and Disk P.
14. Test the failover configuration.

### Usage notes:

To migrate an instance to run in an MSCS failover environment, you need to migrate the instance on the current machine first, then add other MSCS nodes to the instance using the **db2iclus** with the ADD option.

To revert an MSCS instance back to a regular instance, you first need to remove all other MSCS nodes from the instance by using the **db2iclus** with the DROP option. Next, you should undo the migration for the instance on the current machine.

### Related reference:

- "db2icrt - Create instance " on page 122
- "db2idrop - Remove instance" on page 125
- "db2imigr - Migrate instance " on page 128
- "db2stop - Stop DB2 " on page 272

## db2icrt - Create instance

Creates DB2 instances.

On Linux and UNIX-based systems, this utility is located in the DB2DIR/instance directory, where DB2DIR represents the installation location where the current version of the DB2 database system is installed. On Windows operating systems, this utility is located under the DB2PATH\bin directory where DB2PATH is the location where the DB2 copy is installed.

The **db2icrt** command creates DB2 instances related to the DB2 copy installation path from where **db2icrt** is issued.

### Authorization:

Root access on Linux and UNIX-based systems or Local Administrator authority on Windows operating systems.

### Command syntax:

#### For Linux and UNIX-based systems

```

▶▶ db2icrt [-h] [-d] [-a AuthType] [-p PortName]
 [-s InstType] [-u FencedID] InstName

```

#### For Windows operating systems

```

▶▶ db2icrt InstName [-s InstType] [-u UserName, Password]
 [-p InstProfPath] [-h HostName] [-r PortRange] [-?]

```

### Command parameters:

#### For Linux and UNIX-based systems

##### -h or -?

Displays the usage information.

**-d** Turns debug mode on. Use this option only when instructed by DB2 Support.

##### -a *AuthType*

Specifies the authentication type (SERVER, CLIENT or SERVER\_ENCRYPT) for the instance. The default is SERVER.

##### -p *PortName*

Specifies the port name or number used by the instance. This option does not apply to client instances.

**-s** *InstType*

Specifies the type of instance to create. Use the -s option only when you are creating an instance other than the default for your system. Valid values are:

**client** Used to create an instance for a client. Use this value if you are using DB2 Connect Personal Edition.

**ese** Used to create an instance for a database server with local and remote clients.

**wse** Used to create an instance for DB2 Workgroup Server Edition, DB2 Express Edition and DB2 Connect Enterprise Edition.

**-u** *Fenced ID*

Specifies the name of the user ID under which fenced user-defined functions and fenced stored procedures will run. The -u option is required if you are not creating a client instance.

*InstName*

Specifies the name of the instance which is also the name of an existing user in the operating system.

**For Windows operating systems***InstName*

Specifies the name of the instance.

**-s** *InstType*

Specifies the type of instance to create. Valid values are:

**client** Used to create an instance for a client. Use this value if you are using DB2 Connect Personal Edition.

**standalone**

Used to create an instance for a database server with local clients.

**ese** Used to create an instance for a database server with local and remote clients.

**wse** Used to create an instance for DB2 Workgroup Server Edition, DB2 Express Edition and DB2 Connect Enterprise Edition.

**-u** *Username, Password*

Specifies the account name and password for the DB2 service. This option is required when creating a partitioned database instance.

**-p** *InstProfPath*

Specifies the instance profile path.

**-h** *HostName*

Overrides the default TCP/IP host name if there is more than one for the current machine. The TCP/IP host name is used when creating the default database partition (database partition 0). This option is only valid for partitioned database instances.

**-r** *PortRange*

Specifies a range of TCP/IP ports to be used by the partitioned database instance when running in MPP mode. For example, -r 50000,50007. The services file of the local machine will be updated with the following entries if this option is specified:

```
DB2_InstName baseport/tcp
DB2_InstName_END endport/tcp
```

## db2icrt - Create Instance

-? Displays usage information.

### Examples:

- On an AIX machine, to create an instance for the user ID db2inst1, issue the following command:

On a client machine:

```
DB2DIR/instance/db2icrt db2inst1
```

On a server machine:

```
DB2DIR/instance/db2icrt -u db2fenc1 db2inst1
```

where db2fenc1 is the user ID under which fenced user-defined functions and fenced stored procedures will run.

### Usage notes:

- The -s option is intended for situations in which you want to create an instance that does not use the full functionality of the system. For example, if you are using Enterprise Server Edition (ESE), but do not want partition capabilities, you could create a Workgroup Server Edition (WSE) instance, using the option -s WSE.
- To create a DB2 instance that supports Microsoft Cluster Server, first create an instance, then use the **db2iclus** command to migrate it to run in a MSCS instance.
- Only once instance can be created under a user name. If you want to create an instance under a user name that already has a related instance, you must drop instance before creating the new one.

### Related concepts:

- “User, user ID and group naming rules” in *Administration Guide: Implementation*

### Related reference:

- “db2iclus - Microsoft cluster server” on page 119

## db2idrop - Remove instance

Removes a DB2 instance that was created by **db2icrt**. You can only drop instances that are listed by **db2ilist** for the same DB2 copy where you are issuing **db2idrop** from.

On Linux and UNIX-based systems, this utility is located in the DB2DIR/instance directory, where DB2DIR represents the installation location where the current version of the DB2 database system is installed. On Windows operating systems, this utility is located under the DB2PATH\bin directory where DB2PATH is the location where the DB2 copy is installed.

### Authorization:

Root access on Linux and UNIX-based systems or Local Administrator on Windows operating systems.

### Command syntax:

#### For Linux and UNIX-based systems

```
▶▶ db2idrop InstName [-d] [-f] [-h] [-?]
```

#### For Windows Operating Systems

```
▶▶ db2idrop InstName [-f] [-h]
```

### Command parameters:

#### For Linux and UNIX-based systems

##### InstName

Specifies the name of the instance.

**-d** Enters debug mode, for use by DB2 Service.

**-f** Specifies the force applications flag. If this flag is specified all the applications using the instance will be forced to terminate.

##### **-h or -?**

Displays the usage information.

#### For Windows Operating Systems

##### InstName

Specifies the name of the instance.

**-f** Specifies the force applications flag. If this flag is specified all the applications using the instance will be forced to terminate.

**-h** Displays usage information.

### Examples:

- If you created db2inst1 on a Linux and UNIX-based system by issuing the following command:

## db2idrop - Remove Instance

```
/opt/IBM/db2/copy1/instance/db2icrt -u db2fenc1 db2inst1
```

To drop db2inst1, you must run the following command:

```
/opt/IBM/db2/copy1/instance/db2idrop db2inst1
```

### Usage notes:

- In a partitioned database environment, if more than one database partition belongs to the instance that is being dropped, the **db2idrop** command has to be run on each database partition so that the DB2 registry on each database partition is updated.
- Before an instance is dropped, ensure that the DB2 database manager has been stopped and that DB2 database applications accessing the instance are disconnected and terminated. DB2 databases associated with the instance can be backed up, and configuration data saved for future reference if needed.
- The **db2idrop** command does not remove any databases. Please remove the databases first if they are no longer required. If the databases are not removed, they can always be catalogued under another DB2 copy of the same release and continued to be used.

### Related reference:

- “db2icrt - Create instance ” on page 122



---

## db2ilist - List instances

Lists all the instances that are created using the **db2icrt** command from the same DB2 copy location that you are running the **db2ilist** command.

On Linux and UNIX-based systems, this utility is located in the DB2DIR/instance directory, where DB2DIR is the instance directory where the DB2 copy is installed. On Windows operating systems, this utility is located under the DB2PATH\bin directory where DB2PATH represents the installation location where the current version of the DB2 database system is installed.

### Authorization:

None

### Command syntax:

```
▶▶ db2ilist [-h]
```

### Command parameters:

**-h** Displays usage information.

### Related tasks:

- “Removing instances” in *Administration Guide: Implementation*
- “Setting up the DB2 administration server (DAS) to use the Configuration Assistant and the Control Center” in *Administration Guide: Implementation*
- “Updating instance configuration on UNIX” in *Administration Guide: Implementation*

### Related reference:

- “db2iupdt - Update instances ” on page 135

## db2imigr - Migrate instance

On Linux and UNIX-based systems, migrates an instance from a previous version of DB2 database system (supported for migration to the current version of DB2 database system) to the current version of the DB2 copy from where you are running the **db2imigr** command. This utility is located in the DB2DIR/instance directory, where DB2DIR represents the installation location where the current version of the DB2 database system is installed.

On Windows operating systems, migrates an instance from a past release to the current release. Execute the **db2imigr** command from the DB2 copy that you want to migrate the instance to. To move your instance profile from its current location to another location, use the /p option and specify the instance profile path. Otherwise, the instance profile will stay in its original location after migration. This utility is located in the DB2PATH\bin directory, where DB2PATH is the location where the DB2 copy is installed.

The **db2imigr** command calls the **db2ckmig** command to verify that databases are ready for migration before migrating the instance. The migration will not continue if the **db2ckmig** command returns any errors.

### Authorization:

Root access on Linux and UNIX-based systems or Local Administrator on Windows operating systems.

### Command syntax:

#### For Linux and UNIX-based systems

```
db2imigr [-d] [-a AuthType] [-u FencedID] InstName
```

#### For Windows operating systems

```
db2imigr InstName /u:username,password [/p:instance-profile-path]
[/q] [/a:authType] [/?]
```

### Command parameters:

#### For Linux and UNIX-based systems

- d** Turns debug mode on. Use this option only when instructed by DB2 Support.
- a AuthType** Specifies the authentication type (SERVER, CLIENT or SERVER\_ENCRYPT) for the instance. The default is SERVER.
- u FencedID** Specifies the name of the user ID under which fenced user-defined functions and fenced stored procedures will run. This option is optional if a DB2 client only is installed.

*InstName*

Specifies the name of the instance.

### For Windows operating systems

**InstName**

Specifies the name of the instance.

*/u:username,password*

Specifies the account name and password for the DB2 service. This option is required when migrating a partitioned instance.

*/p:instance-profile-path*

Specifies the new instance profile path for the migrated instance.

*/q* Issues the **db2imigr** command in quiet mode.

*/a:authType*

Specifies, *authType*, the authentication type (SERVER, CLIENT, or SERVER\_ENCRYPT) for the instance .

*/?* Displays usage information for the **db2imigr** command.

### Usage notes:

#### For Linux and UNIX-based systems

- The **db2imigr** command removes any symbolic links that exist in `/usr/lib` and `/usr/include` in version you are migrating from. If you have applications that load `libdb2` directly from `/usr/lib` rather than using the operating system's library environment variable to find it, your applications might fail to execute properly after you have run **db2imigr**.
- If you use the **db2imigr** command to migrate a DB2 instance from a previous version to the current version of a DB2 database system, the DB2 Global Profile Variables defined in an old DB2 database installation path will not be migrated over to the new installation location. The DB2 Instance Profile Variables specific to the instance to be migrated will be carried over after the instance is migrated.

### Related concepts:

- "Migration overview for DB2 servers" in *Migration Guide*

### Related tasks:

- "Migrating instances" in *Migration Guide*

### Related reference:

- "dasmigr - Migrate the DB2 administration server " on page 6
- "db2ckmig - Database pre-migration tool " on page 61

---

### db2inidb - Initialize a mirrored database

Initializes a mirrored database in a split mirror environment. The mirrored database can be initialized as a clone of the primary database, placed in roll forward pending state, or used as a backup image to restore the primary database. This command can only be run against a split mirror database, and it must be run before the split mirror can be used.

#### Authorization:

One of the following:

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

#### Required connection:

None

#### Command syntax:

```
db2inidb database_alias AS [SNAPSHOT | STANDBY | MIRROR] RELOCATE USING configFile
```

#### Command parameters:

##### database\_alias

Specifies the alias of the database to be initialized.

##### SNAPSHOT

Specifies that the mirrored database will be initialized as a clone of the primary database.

##### STANDBY

Specifies that the database will be placed in roll forward pending state. New logs from the primary database can be fetched and applied to the standby database. The standby database can then be used in place of the primary database if it goes down.

##### MIRROR

Specifies that the mirrored database is to be used as a backup image which can be used to restore the primary database.

##### RELOCATE USING configFile

Specifies that the database files are to be relocated based on the information listed in the specified *configFile* prior to initializing the database as a snapshot, standby, or mirror. The format of *configFile* is described in `db2relocatedb - Relocate database command`.

#### Usage notes:

Do not issue the **db2 connect to <database>** command before issuing the **db2init <database> as mirror** command. Attempting to connect to a split mirror database before initializing it erases the log files needed during roll forward recovery. The connect sets your database back to the state it was in when you suspended the database. If the database is marked as consistent when it was suspended, the DB2

database system concludes there is no need for crash recovery and empties the logs for future use. If the logs have been emptied, attempting to roll forward results in the SQL4970N error message being returned.

In a partitioned database environment, **db2inidb** must be run on every database partition before the split mirror from any of the database partitions can be used. **db2inidb** can be run on all database partitions simultaneously using the **db2\_all** command.

If, however, you are using the RELOCATE USING option, you cannot use the **db2\_all** command to run **db2inidb** on all of the partitions simultaneously. A separate configuration file must be supplied for each partition, that includes the NODENUM value of the database partition being changed. For example, if the name of a database is being changed, every database partition will be affected and the **db2relocatedb** command must be run with a separate configuration file on each database partition. If containers belonging to a single database partition are being moved, the **db2relocatedb** command only needs to be run once on that database partition.

If the RELOCATE USING *configFile* parameter is specified and the database is relocated successfully, the specified *configFile* will be copied into the database directory and renamed to *db2path.cfg*. During a subsequent crash recovery or rollforward recovery, this file will be used to rename container paths as log files are being processed.

If a clone database is being initialized, the specified *configFile* will be automatically removed from the database directory after a crash recovery is completed.

If a standby database or mirrored database is being initialized, the specified *configFile* will be automatically removed from the database directory after a rollforward recovery is completed or canceled. New container paths can be added to the *db2path.cfg* file after **db2inidb** has been run. This would be necessary when CREATE or ALTER TABLESPACE operations are done on the original database and different paths must be used on the standby database.

### Related tasks:

- “Using a split mirror to clone a database” in *Data Recovery and High Availability Guide and Reference*
- “Using a split mirror as a backup image” in *Data Recovery and High Availability Guide and Reference*
- “Using a split mirror as a standby database” in *Data Recovery and High Availability Guide and Reference*

### Related reference:

- “db2relocatedb - Relocate database ” on page 235
- “rah and db2\_all command descriptions” in *Administration Guide: Implementation*

---

### db2inspf - Format inspect results

This utility formats the data from INSPECT CHECK results into ASCII format. Use this utility to see details of the inspection. The formatting by the db2inspf utility can be format for a table only, or a table space only, or for errors only, or warnings only, or summary only.

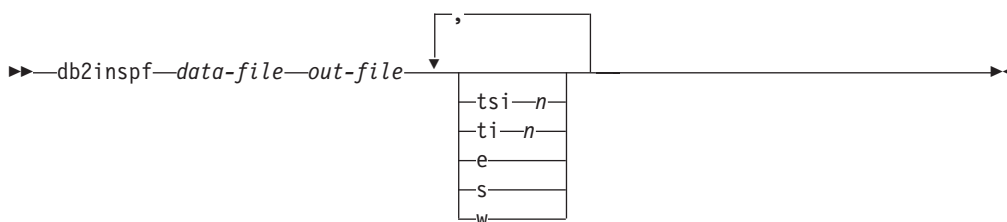
#### Authorization:

Anyone can access the utility, but users must have read permission on the results file in order to execute this utility against them.

#### Required connection:

None

#### Command syntax:



#### Command Parameters:

##### data-file

The unformatted inspection results file to format.

##### out-file

The output file for the formatted output.

- tsi n** Table space ID. Format out only for tables in this table space.
- ti n** Table ID. Format out only for table with this ID, table space ID must also be provided.
- e** Format out errors only.
- s** Summary only.
- w** Warnings only.

#### Related reference:

- "INSPECT " on page 512
- "db2Inspect API - Inspect database for architectural integrity" in *Administrative API Reference*

## db2isetaup - Start instance creation interface

Starts the DB2 Instance Setup wizard, a graphical tool for creating instances and for configuring new functionality on existing instances.

### Authorization:

Root authority on the system where the command is issued.

### Required connection:

None.

### Command syntax:

```

▶▶ db2isetaup [-i language-code] [-l logfile] [-t tracefile]
 [-r response_file] [--? -h]

```

### Command parameters:

- i** *language-code*  
Two letter code for the preferred language in which to run the install. If unspecified, this parameter will default to the locale of the current user.
- l** *logfile*  
Full path and name of the log file. If no name is specified, the path and filename default to /tmp/db2isetaup.log
- t** *tracefile*  
The full path and name of trace file specified by *tracefile*.
- r** *response\_file*  
Full path and file name of the response file to use.
- , -h** Output usage information.

### Usage notes:

1. This instance setup wizard provides a subset of the functionality provided by the DB2 Setup wizard. The DB2 Setup wizard (which runs from the installation media) allows you to install DB2 components, do system setup tasks such as DAS creation/configuration, and set up instances. The DB2 Instance Setup wizard only provides the functionality pertaining to instance setup.
2. The executable file for this command is located in the DB2DIR/instance directory, along with other instance scripts such as db2icrt and db2iupdt. DB2DIR represents the installation location where the current version of the DB2 database system is installed. Like these other instance scripts, it requires root authority. It is available in a typical install, but not in a compact install.
3. db2isetaup runs on all supported Linux and UNIX-based systems.

### Related concepts:

- "DB2 installation methods" in *Quick Beginnings for DB2 Servers*

### Related reference:

## db2iseta - Start Instance Creation Interface

- "db2iupdt - Update instances " on page 135
- "db2setu - Install DB2" on page 248
- "db2icrt - Create instance " on page 122



## db2iupdt - Update instances

On Linux and UNIX-based systems, this command updates a specified DB2 instance. The **db2iupdt** command can be issued against instances of the same version that are associated with the same, or a different DB2 database installation directory. In all cases, it will update the instance so that it runs against the code located in the same directory as where you issued the **db2iupdt** command. You should issue this command:

- whenever you install a new DB2 database product or Fix Pack to the installation directory related to the DB2 instance.
- if you want to bring a DB2 instance from one installation path to the current one for the same version of DB2 database system.

On Linux and UNIX-based systems, it is located in the DB2DIR/instance directory, where DB2DIR is the location where the current version of the DB2 database product is installed.

On Windows operating systems, this command updates the instance release level. It can also be used to move an instance from one DB2 copy to another. The instance is moved to the DB2 copy you execute **db2iupdt** from. To move your instance profile from its current location to another location, use the /p option and specify the instance profile path. Otherwise, the instance profile will stay in its original location after update. Use the **db2imigr** command instead to change from a major release to another. This utility is located in the DB2PATH\sql1lib\bin directory, where DB2PATH is the location where the current version of the DB2 database product is installed.

To update an instance with **db2iupdt**, you must first stop all processes that are running for the instance.

### Authorization:

Root access on UNIX operating systems or Local Administrator on Windows operating systems.

### Command syntax:

#### For UNIX operating systems

```

▶▶ db2iupdt [-h] [-d] [-k] [-D] [-s] [-a AuthType]
 [-u FencedID] [-e InstName]

```

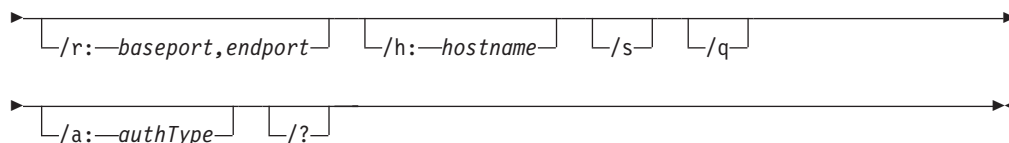
#### For Windows operating systems

```

▶▶ db2iupdt InstName /u:—username,password [-p:—instance-profile-path]

```

## db2iupdt - Update Instances



### Command parameters:

#### For UNIX operating systems

##### -h or -?

Displays the usage information.

##### -d

Turns debug mode on.

##### -k

Keeps the current instance type during the update.

##### -D

Moves an instance from a higher code level on one path to a lower code level installed on another path.

##### -s

Ignores the existing SPM log directory.

##### -a *AuthType*

Specifies the authentication type (SERVER, SERVER\_ENCRYPT or CLIENT) for the instance. The default is SERVER.

##### -u *Fenced ID*

Specifies the name of the user ID under which fenced user defined functions and fenced stored procedures will run. This option is only needed when converting an instance from a client instance to a server instance type. If an instance is already a server instance, or if an instance is a client instance and is staying as a client instance (by using the -k option), the -u option is not needed. The -u option cannot change the fenced user for an existing instance.

##### **InstName**

Specifies the name of the instance.

##### -e

Updates every instance.

#### For Windows operating systems

##### **InstName**

Specifies the name of the instance.

##### */u:username,password*

Specifies the account name and password for the DB2 service.

##### */p:instance-profile-path*

Specifies the new instance profile path for the updated instance.

##### */r:baseport,endport*

Specifies the range of TCP/IP ports to be used by the partitioned database instance when running in MPP mode. When this option is specified, the services file on the local machine will be updated with the following entries:

```
DB2_InstName baseport/tcp
DB2_InstName_END endport/tcp
```

##### */h:hostname*

Overrides the default TCP/IP host name if there are more than one TCP/IP host names for the current machine.

##### /s

Updates the instance to a partitioned instance.

**/q** Issues the **db2iupdt** command in quiet mode.

**/a:authType**

Specifies, **authType**, the authentication type (SERVER, CLIENT, or SERVER\_ENCRYPT) for the instance .

**/?** Displays usage information for the **db2iupdt** command.

### Examples (UNIX):

1. If you have an instance **db2inst1** related to the installation path **DB2DIR** and you applied a Fix Pack on top of the installation path, you may need to update the instance by running the following command:

```
<DB2DIR>/instance/db2iupdt db2inst1
```

This will bring up the instance to the highest type of instance. To keep the original instance type, you might need to use the **-k** option.

2. An instance, **db2inst2**, is related to the installation path **DB2DIR1**. You have another installation of the DB2 database product on the same system at **DB2DIR2** for the same version of the DB2 database product as that installed on **DB2DIR1**. To update the instance to use the installed DB2 database product from **DB2DIR1** to **DB2DIR2**, issue the following command:

```
<DB2DIR2>/instance/db2iupdt db2inst2
```

If the DB2 database product installed at **DB2DIR2** is at level lower than that at **DB2DIR1**, issue:

```
<DB2DIR2>/instance/db2iupdt -D db2inst2
```

### Usage notes:

#### For UNIX operating systems

- If you use the **db2iupdt** command to update a DB2 instance from another installation location to the current installation location, the DB2 Global Profile Variables defined in an old DB2 database installation path will not be updated over to the new installation location. The DB2 Instance Profile Variables specific to the instance to be updated will be carried over after the instance is updated.

### Related tasks:

- “Updating instance configuration on Windows” in *Administration Guide: Implementation*
- “Updating instance configuration on UNIX” in *Administration Guide: Implementation*

### Related reference:

- “db2ilist - List instances ” on page 127

## db2jdbcbind - DB2 JDBC package binder

This utility is used to bind or rebind the JDBC packages to a DB2 database. DB2 Version 8 databases already have the JDBC packages preinstalled, therefore, this command is usually necessary only for downlevel servers. JDBC and CLI share the same packages. If the CLI packages have already been bound to a database, then it is not necessary to run this utility and vice versa.

### Authorization:

One of the following:

- *sysadm*
- *dbadm*
- 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

### Required connection:

This command establishes a database connection.

### Command syntax:

```
▶ db2jdbcbind help -url jdbc:db2://server:port/dbname --user username
▶ --password password --collection collection ID
▶ --size number of packages
▶ --tracelevel TRACE_ALL, TRACE_CONNECTION_CALLS, TRACE_CONNECTS, TRACE_DIAGNOSTICS, TRACE_DRDA_FLOWS, TRACE_DRIVER_CONFIGURATION, TRACE_NONE, TRACE_PARAMETER_META_DATA, TRACE_RESULT_SET_CALLS, TRACE_RESULT_SET_META_DATA, TRACE_STATEMENT_CALLS
```

### Command parameters:

## db2jdbcbind - DB2 JDBC Package Binder Utility

- help** Displays help information, all other options are ignored.
- url jdbc:db2://server:port/dbname**  
Specifies a JDBC URL for establishing the database connection. The DB2 JDBC type 4 driver is used to establish the connection.
- user *username***  
Specifies the name used when connecting to a database.
- password *password***  
Specifies the password for the user name.
- collection *collection ID***  
The collection identifier (CURRENT PACKAGESET), to use for the packages. The default is NULLID. Use this to create multiple instances of the package set. This option can only be used in conjunction with the Connection or DataSource property currentPackageSet.
- size *number of packages***  
The number of internal packages to bind for each DB2 transaction isolation level and holdability setting. The default is 3. Since there are four DB2 isolation levels and two cursor holdability settings, there will be  $4 \times 2 = 8$  times as many dynamic packages bound as are specified by this option. In addition, a single static package is always bound for internal use.
- tracelevel**  
Identifies the level of tracing, only required for troubleshooting.

---

### db2ldcfg - Configure LDAP environment

Configures the Lightweight Directory Access Protocol (LDAP) user distinguished name (DN) and password for the current logon user in an LDAP environment using an IBM LDAP client.

#### Authorization:

*none*

#### Required connection:

None

#### Command syntax:

```
▶▶ db2ldcfg -u user's Distinguished Name -w password ▶▶
```

#### Command parameters:

##### -u user's Distinguished Name

Specifies the LDAP user's Distinguished Name to be used when accessing the LDAP directory. As shown in the example below, the Distinguished name has several parts: the user ID, such as `jdoue`, the domain and organization names and the suffix, such as `com` or `org`.

##### -w password

Specifies the password.

**-r** Removes the user's DN and password from the machine environment.

#### Example:

```
db2ldcfg -u "uid=jdoue,dc=mydomain,dc=myorg,dc=com" -w password
```

#### Usage notes:

In an LDAP environment using an IBM LDAP client, the default LDAP user's DN and password can be configured for the current logon user. Once configured, the LDAP user's DN and password are saved in the user's environment and used whenever DB2 accesses the LDAP directory. This eliminates the need to specify the LDAP user's DN and password when issuing the LDAP command or API. However, if the LDAP user's DN and password are specified when the command or API is issued, the default settings will be overridden.

This command can only be run when using an IBM LDAP client. On a Microsoft LDAP client, the current logon user's credentials will be used.

#### Related tasks:

- "Configuring the LDAP user for DB2 applications" in *Administration Guide: Implementation*

#### Related reference:

- "CATALOG LDAP DATABASE " on page 377

---

## db2level - Show DB2 service level

Shows the current Version and Service Level of the installed DB2 product. Output from this command goes to the console by default.

### Authorization:

None.

### Required Connection:

None.

### Command Syntax:

▶▶—db2level—▶▶

### Examples:

On Windows operating systems, the **db2level** command shows the DB2 copy name. For example:

```
DB21085I Instance "DB2" uses "32" bits and DB2 code release "SQL09010" with
level identifier "01010107".
Informational tokens are "DB2 v9.1.0.189", "n060119", "", and Fix Pack "0".
Product is installed at "c:\SQLLIB" with DB2 Copy Name "db2build".
```

On Linux and UNIX based operating systems, the **db2level** command does not show the DB2 copy name. For example:

```
DB21085I Instance "wqzhuang" uses "64" bits and DB2 code release "SQL09010"
with level identifier "01010107".
Informational tokens are "DB2 v9.1.0.0", "n060124", "", and Fix Pack "0".
Product is installed at "/home/wqzhuang/sqllib".
```

The information output by the command includes Release, Level, and various informational tokens.

### Related concepts:

- “Identifying the version and service level of your product” in *Troubleshooting Guide*

### Related reference:

- “ENV\_INST\_INFO administrative view – Retrieve information about the current instance” in *Administrative SQL Routines and Views*

## db2licm - License management tool

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

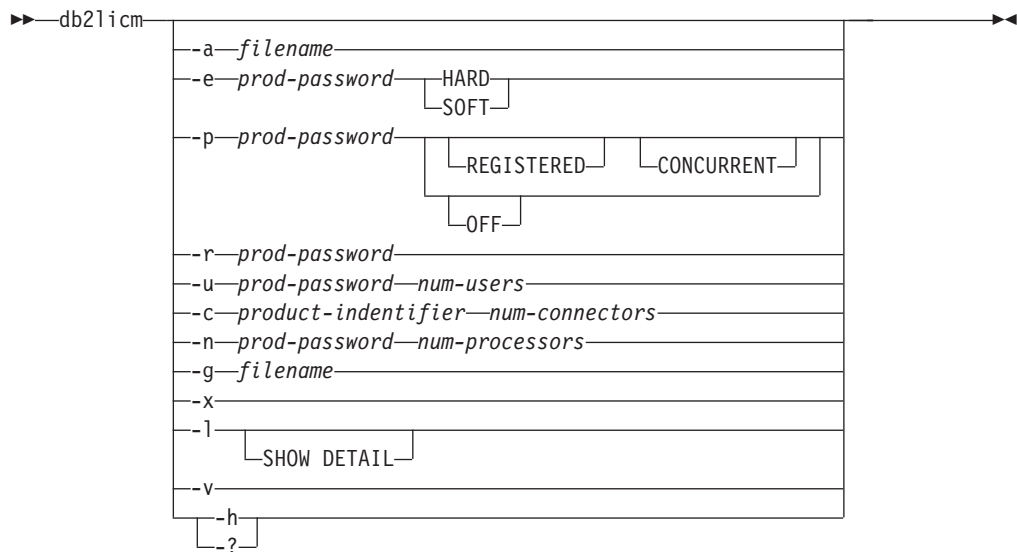
**Authorization:**

*sysadm* for license removal only.

**Required connection:**

None

**Command syntax:**



**Command parameters:**

**-a filename**

Adds a license for a product. Specify a file name containing valid license information. This can be obtained from your licensed product CD or by contacting your IBM representative or authorized dealer.

**-e product-identifier**

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 unlicensed requests will be logged but not restricted.

**-p product-identifier keyword**

Updates the license policy type to use on the system. Specify OFF to turn off all policies.

**-r product-identifier**

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 product-identifier num-users**

Updates the number of user licenses that the customer has purchased. Specify the product identifier and the number of users.



**-c product-identifier num-connectors**

Updates the number of connector entitlements that have been purchased. Specify the product identifier and the number of connector entitlements.

**-n product-identifier num-processors**

Updates the number of entitled processors. Specify the product identifier and the number of processors that you are entitled to use this product with.

**-g filename**

Generates compliance report. Specify file name where output is to be stored.

**-x** Resets license compliance information for the purposes of license compliance report.

**-l** Lists all the products with available license information, including the product identifier. Specify SHOW DETAIL to view detailed information about licensed features (if any).

**-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 db2ese.lic
db2licm -p db2wse registered concurrent
db2licm -r db2ese
db2licm -u db2wse 10
db2licm -n db2ese 8
```

**Related tasks:**

- “Registering a DB2 product or feature license key using the db2licm command” in *Installation and Configuration Supplement*

### db2listvolumes - Display GUIDs for all disk volumes

Displays the GUIDs for all the disk volumes defined on a Windows operating system. This command creates two files in the directory where the tool is issued from. One file, called `volumes.xml`, contains information about each disk volume encoded in XML for easy viewing on an XML-enabled browser. The second file, called `tablespace.ddl`, contains the required syntax for specifying table space containers. This file must be updated to fill in the remaining information needed for a table space definition. The **db2listvolumes** command does not require any command line arguments. It is only available on Windows operating systems.

**Authorization:**

Administrator

**Required Connection:**

None.

**Command syntax:**

▶▶—db2listvolumes—▶▶

**Command parameters:**

None.

**Related tasks:**

- “Attaching a direct disk access device” in *Administration Guide: Implementation*

---

### db2logsforrfd - List logs required for rollforward recovery

Parses the DB2TSCHG.HIS file. This utility allows a user to find out which log files are required for a table space rollforward operation. This utility is located in `sql1lib/bin`.

**Authorization:**

**Required connection:**

None.

**Command syntax:**

```
▶▶—db2logsforrfd—path [—all]▶▶
```

**Command parameters:**

*path* Full path and name of the DB2TSCHG.HIS file.

**-all** Displays more detailed information.

**Examples:**

```
db2logsForRfwd /home/ofer/ofer/NODE0000/S0000001/DB2TSCHG.HIS
```

```
db2logsForRfwd DB2TSCHG.HIS -all
```

**Related concepts:**

- “Rolling forward changes in a table space” in *Data Recovery and High Availability Guide and Reference*

## db2look - DB2 statistics and DDL extraction tool

Extracts the required Data Definition Language (DDL) statements to reproduce the database objects of a production database on a test database. The **db2look** command generates the DDL statements by object type.

This tool can generate the required UPDATE statements used to replicate the statistics on the objects in a test database. It can also be used to generate the **UPDATE DATABASE CONFIGURATION** and **UPDATE DATABASE MANAGER CONFIGURATION** commands and the **db2set** commands so that query optimizer-related configuration parameters and registry variables on the test database match those of the 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. Using this tool makes it possible to create a test database where access plans are similar to those that would be used on the production system.

You should check the DDL statements generated by the **db2look** command since they might not exactly reproduce all characteristics of the original SQL objects. For table spaces on partitioned database environments, DDL might not be complete if some database partitions are not active. Make sure all database partitions are active using the **ACTIVATE** command.

### Authorization:

SELECT privilege on the system catalog tables.

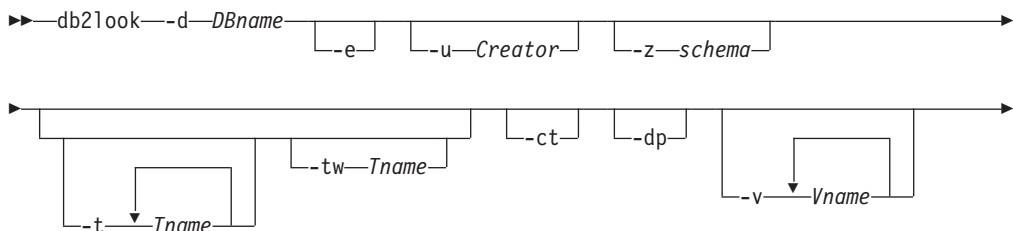
In some cases, such as generating table space container DDL (which calls the APIs `sqlbotcq`, `sqlbftcq`, and `sqlbctcq`), you will require one of the following:

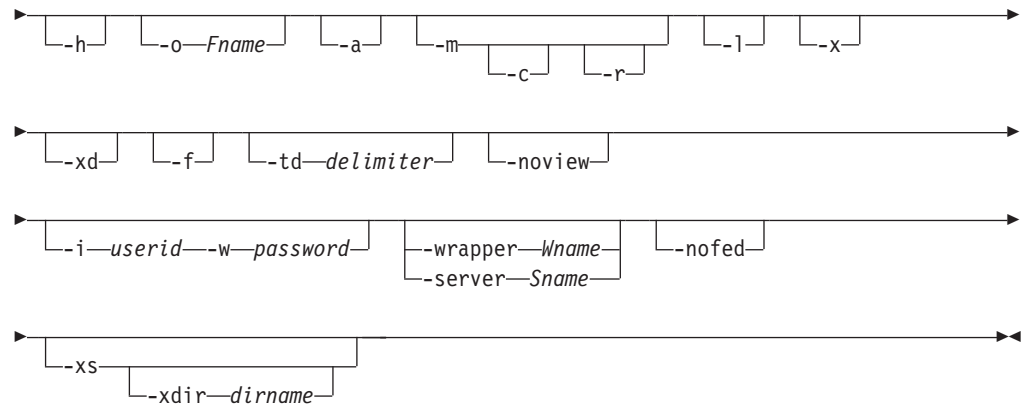
- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

### Required connection:

None

### Command syntax:





### Command parameters:

#### **-d** *DBname*

Alias name of the production database that is to be queried. *DBname* can be the name of a DB2 Database for Linux, UNIX, and Windows or DB2 Version 9.1 for z/OS (DB2 for z/OS) database. If the *DBname* is a DB2 for z/OS database, the **db2look** utility will extract the DDL and UPDATE statistics statements for OS/390 and z/OS objects. These DDL and UPDATE statistics statements are statements applicable to a DB2 Database for Linux, UNIX, and Windows database and not to a DB2 for z/OS database. This is useful for users who want to extract OS/390 and z/OS objects and recreate them in a DB2 Database for Linux, UNIX, and Windows database.

If *DBname* is a DB2 for z/OS database, the output of the **db2look** command is limited to the following:

- Generate DDL for tables, indexes, views, and user-defined distinct types
- Generate UPDATE statistics statements for tables, columns, column distributions and indexes

#### **-e** Extract DDL statements for database objects. DDL for the following database objects are extracted when using the **-e** option:

- Tables
- Views
- Materialized query tables (MQT)
- Aliases
- Indexes
- Triggers
- Sequences
- User-defined distinct types
- Primary key, referential integrity, and check constraints
- User-defined structured types
- User-defined functions
- User-defined methods
- User-defined transforms
- Wrappers
- Servers
- User mappings
- Nicknames

## db2look - DB2 Statistics and DDL Extraction Tool

- Type mappings
- Function templates
- Function mappings
- Index specifications
- Stored procedures

The DDL generated by the **db2look** command can be used to recreate user-defined functions successfully. However, the user source code that a particular user-defined function references (the EXTERNAL NAME clause, for example) must be available in order for the user-defined function to be usable.

### **-u** *Creator*

Creator ID. Limits output to objects with this creator ID. If option **-a** is specified, this parameter is ignored. The output will not include any inoperative objects. To display inoperative objects, use the **-a** option.

### **-z** *schema*

Schema name. Limits output to objects with this schema name. The output will not include any inoperative objects. To display inoperative objects, use the **-a** option. If this parameter is not specified, objects with all schema names are extracted. If the **-a** option is specified, this parameter is ignored. This option is ignored for the federated DDL.

### **-t** *Tname1 Tname2 ... TnameN*

Table name list. Limits the output to particular tables in the table list. The maximum number of tables is 30. Table names are separated by a blank space. Case-sensitive names must be enclosed inside a backward slash and double quotation delimiter, for example, `\ " MyTabLe \ "`. For multiple-word table names, the delimiters must be placed within quotation marks (for example, `"\ "My Table\ ""`) to prevent the pairing from being evaluated word-by-word by the command line processor. If a multiple-word table name is not enclosed by the backward slash and double delimiter (for example, `"My Table"`), all words will be converted into uppercase and the **db2look** command will look for an uppercase table (for example, `"MY TABLE"`). When **-t** is used with **-l**, the combination does not support partitioned tables in DB2 Version 9.1.

### **-tw** *Tname*

Generates DDL for table names that match the pattern criteria specified by *Tname*. Also generates the DDL for all dependent objects of all returned tables. *Tname* can be a single value only. The underscore character (`_`) in *Tname* represents any single character. The percent sign (`%`) represents a string of zero or more characters. Any other character in *Tname* only represents itself. When **-tw** is specified, the **-t** option is ignored.

### **-ct**

Generate DDL by object creation time. Generating DDL by object creation time will not guarantee that all the object DDLs will be displayed in correct dependency order. The **db2look** command only supports the following options if the **-ct** option is also specified: **-e**, **-a**, **-u**, **-z**, **-t**, **-tw**, **-v**, **-l**, **-noview**.

### **-dp**

Generate DROP statement before CREATE statement. The DROP statement might not work if there is an object that depends on the dropped object. For example, dropping a schema will fail if there is a table that depends on the dropped schema, or dropping a user-defined type/function will fail if there is any other type, function, trigger, or table that depends on it. For typed tables, the DROP TABLE HIERARCHY statement will be generated

for the root table only. A DROP statement is not generated for index, primary and foreign keys, and constraints, because they are always dropped when the table is dropped. When a table has the RESTRICT ON DROP attribute, it cannot be dropped.

- v** *Vname1 Vname2 ... VnameN*  
Generates DDL for the specified views. The maximum number of views is 30. If the **-t** option is specified, the **-v** option is ignored.
- h** Display help information. When this option is specified, all other options are ignored, and only the help information is displayed.
- o** *Fname*  
If using LaTeX format, write the output to *filename.tex*. If using plain text format, write the output to *filename.txt*. Otherwise, write the output to *filename.sql*. If this option is not specified, output is written to standard output. If a filename is specified with an extension, the output will be written into that file.
- a** When this option is specified the output is not limited to the objects created under a particular creator ID. All objects, including inoperative objects, created by all users are considered. For example, if this option is specified with the **-e** option, DDL statements are extracted for all objects in the database. If this option is specified with the **-m** option, UPDATE statistics statements are extracted for all user created tables and indexes in the database. If neither **-u** nor **-a** is specified, the environment variable USER is used. On UNIX operating systems, this variable does not have to be explicitly set; on Windows systems, however, there is no default value for the USER environment variable: a user variable in the SYSTEM variables must be set, or a set USER=<username> must be issued for the session.
- m** Generates the required UPDATE statements to replicate the statistics on tables, statistical views, columns and indexes.
  - c** When this option is specified in conjunction with the **-m** option, the **db2look** command does not generate COMMIT, CONNECT and CONNECT RESET statements. The default action is to generate these statements.
  - r** When this option is specified in conjunction with the **-m** option, the **db2look** command does not generate the **RUNSTATS** command. The default action is to generate the **RUNSTATS** command.
- l** If this option is specified, then the **db2look** command will generate DDL for user defined table spaces, database partition groups and buffer pools. DDL for the following database objects is extracted when using the **-l** option:
  - User-defined table spaces
  - User-defined database partition groups
  - User-defined buffer pools
- x** If this option is specified, the **db2look** command will generate authorization DDL (GRANT statement, for example).  
The supported authorizations include:
  - Table: ALTER, SELECT, INSERT, DELETE, UPDATE, INDEX, REFERENCE, CONTROL
  - View: SELECT, INSERT, DELETE, UPDATE, CONTROL

## db2look - DB2 Statistics and DDL Extraction Tool

- Index: CONTROL
  - Schema: CREATEIN, DROPIN, ALTERIN
  - Database: CREATEDB, BINDADD, CONNECT, CREWATE\_NOT\_FENCED, IMPLICIT\_SCHEMA
  - User-defined function (UDF): EXECUTE
  - User-defined method: EXECUTE
  - Stored procedure: EXECUTE
  - Package: CONTROL, BIND, EXECUTE
  - Column: UPDATE, REFERENCES
  - Table space: USE
  - Sequence: USAGE, ALTER
- xd** If this option is specified, the **db2look** command will generate all authorization DDL including authorization DDL for objects whose authorizations were granted by SYSIBM at object creation time.
- f** Use this option to extract the configuration parameters and registry variables that affect the query optimizer.

The **db2look** command generates an update command for the following configuration parameters:

- Database manager configuration parameters
  - cpuspeed
  - intra\_parallel
  - comm\_bandwidth
  - nodetype
  - federated
  - fed\_noauth
- Database configuration parameters
  - locklist
  - dft\_degree
  - maxlocks
  - avg\_appls
  - stmtheap
  - dft\_queryopt

The **db2look** command generates the **db2set** command for the following DB2 registry variables:

- DB2\_PRED\_FACTORIZE
- DB2\_CORRELATED\_PREDICATES
- DB2\_LIKE\_VARCHAR
- DB2\_SORT\_AFTER\_TQ
- DB2\_ORDERED\_NLJN
- DB2\_NEW\_CORR\_SQ\_FF
- DB2\_PART\_INNER\_JOIN
- DB2\_INTERESTING\_KEYS

**-td** *delimiter*

Specifies the statement delimiter for SQL statements generated by the **db2look** command. If this option is not specified, the default is the



semicolon (;). It is recommended that this option be used if the `-e` option is specified. In this case, the extracted objects might contain triggers or SQL routines.

**-noview**

If this option is specified, CREATE VIEW DDL statements will not be extracted.

**-i** *userid*

Use this option when working with a remote database.

**-w** *password*

Used with the `-i` option, this parameter allows the user to run the **db2look** command against a database that resides on a remote system. The user ID and the password are used by the **db2look** command to log on to the remote system. If working with remote databases, the remote database must be the same version as the local database. The **db2look** command does not have down-level or up-level support.

**-wrapper** *Wname*

Generates DDL statements for federated objects that apply to this wrapper. The federated DDL statements that might be generated include: CREATE WRAPPER, CREATE SERVER, CREATE USER MAPPING, CREATE NICKNAME, CREATE TYPE MAPPING, CREATE FUNCTION ... AS TEMPLATE, CREATE FUNCTION MAPPING, CREATE INDEX SPECIFICATION, and GRANT (privileges to nicknames, servers, indexes). Only one wrapper name is supported; an error is returned if less than one or more than one is specified. This option does not support non-relational data sources.

**-server** *Sname*

Generates DDL statements for federated objects that apply to this server. The federated DDL statements that might be generated include: CREATE WRAPPER, CREATE SERVER, CREATE USER MAPPING, CREATE NICKNAME, CREATE TYPE MAPPING, CREATE FUNCTION ... AS TEMPLATE, CREATE FUNCTION MAPPING, CREATE INDEX SPECIFICATION, and GRANT (privileges to nicknames, servers, indexes). Only one server name is supported; an error is returned if less than one or more than one is specified. This option does not support non-relational data sources.

**-nofed** Specifies that no federated DDL statements will be generated. When this option is specified, the `-wrapper` and `-server` options are ignored.

**-xs** Exports all files necessary to register XML schemas and DTDs at the target database, and generates appropriate commands for registering them. The set of XSR objects that will be exported is controlled by the `-u`, `-z`, and `-a` options.

**-xdir** *dirname*

Places exported XML-related files into the given path. If this option is not specified, all XML-related files will be exported into the current directory.

**Examples:**

- Generate the DDL statements for objects created by user *walid* in database DEPARTMENT. The **db2look** output is sent to file `db2look.sql`:  

```
db2look -d department -u walid -e -o db2look.sql
```

## db2look - DB2 Statistics and DDL Extraction Tool

- Generate the DDL statements for objects that have schema name *ianhe*, created by user *walid*, in database DEPARTMENT. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -u walid -z ianhe -e -o db2look.sql
```

- Generate the UPDATE statements to replicate the statistics for the database objects created by user *walid* in database DEPARTMENT. The output is sent to file `db2look.sql`:

```
db2look -d department -u walid -m -o db2look.sql
```

- Generate both the DDL statements for the objects created by user *walid* and the UPDATE statements to replicate the statistics on the database objects created by the same user. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -u walid -e -m -o db2look.sql
```

- Generate the DDL statements for objects created by all users in the database DEPARTMENT. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -a -e -o db2look.sql
```

- Generate the DDL statements for all user-defined database partition groups, buffer pools and table spaces. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -l -o db2look.sql
```

- Generate the UPDATE statements for optimizer-related database and database manager configuration parameters, as well as the **db2set** statements for optimizer-related registry variables in database DEPARTMENT. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -f -o db2look.sql
```

- Generate the DDL for all objects in database DEPARTMENT, the UPDATE statements to replicate the statistics on all tables and indexes in database DEPARTMENT, the GRANT authorization statements, the UPDATE statements for optimizer-related database and database manager configuration parameters, the **db2set** statements for optimizer-related registry variables, and the DDL for all user-defined database partition groups, buffer pools and table spaces in database DEPARTMENT. The output is sent to file `db2look.sql`.

```
db2look -d department -a -e -m -l -x -f -o db2look.sql
```

- Generate all authorization DDL statements for all objects in database DEPARTMENT, including the objects created by the original creator. (In this case, the authorizations were granted by SYSIBM at object creation time.) The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -xd -o db2look.sql
```

- Generate the DDL statements for objects created by all users in the database DEPARTMENT. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -a -e -td % -o db2look.sql
```

The output can then be read by the CLP:

```
db2 -td% -f db2look.sql
```

- Generate the DDL statements for objects in database DEPARTMENT, excluding the CREATE VIEW statements. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -e -noview -o db2look.sql
```

- Generate the DDL statements for objects in database DEPARTMENT related to specified tables. The **db2look** output is sent to file `db2look.sql`:

```
db2look -d department -e -t tab1 \"My TaB1E2\" -o db2look.sql
```

## db2look - DB2 Statistics and DDL Extraction Tool

- Generate the DDL statements for all objects (federated and non-federated) in the federated database FEDDEPART. For federated DDL statements, only those that apply to the specified wrapper, FEDWRAP, are generated. The **db2look** output is sent to standard output:

```
db2look -d feddepart -e -wrapper fedwrap
```

- Generate a script file that includes only non-federated DDL statements. The following system command can be run against a federated database (FEDDEPART) and yet only produce output like that found when run against a database which is not federated. The **db2look** output is sent to a file out.sql:

```
db2look -d feddepart -e -nofed -o out
```

- Generate the DDL statements for objects that have schema name *walid* in the database DEPARTMENT. The files required to register any included XML schemas and DTDs are exported to the current directory. The **db2look** output is sent to file db2look.sql:

```
db2look -d department -z walid -e -xs -o db2look.sql
```

- Generate the DDL statements for objects created by all users in the database DEPARTMENT. The files required to register any included XML schemas and DTDs are exported to directory /home/ofer/ofer/. The **db2look** output is sent to standard output:

```
db2look -d department -a -e -xs -xdir /home/ofer/ofer/
```

### Usage notes:

On Windows operating systems, the **db2look** command must be run from a DB2 command window.

Several of the existing options support a federated environment. The following **db2look** command line options are used in a federated environment:

- -e  
When used, federated DDL statements are generated.
- -x  
When used, GRANT statements are generated to grant privileges to the federated objects.
- -xd  
When used, federated DDL statements are generated to add system-granted privileges to the federated objects.
- -f  
When used, federated-related information is extracted from the database manager configuration.
- -m  
When used, statistics for nicknames are extracted.

The ability to use federated systems needs to be enabled in the database manager configuration in order to create federated DDL statements. After the **db2look** command generates the script file, you must set the *federated* configuration parameter to YES before running the script.

You need to modify the output script to add the remote passwords for the CREATE USER MAPPING statements.

## db2look - DB2 Statistics and DDL Extraction Tool

You need to modify the **db2look** command output script by adding **AUTHORIZATION** and **PASSWORD** to those **CREATE SERVER** statements that are used to define a DB2 family instance as a data source.

Usage of the **-tw** option is as follows:

- To both generate the DDL statements for objects in the DEPARTMENT database associated with tables that have names beginning with abc and send the output to the db2look.sql file:

```
db2look -d department -e -tw abc% -o db2look.sql
```

- To generate the DDL statements for objects in the DEPARTMENT database associated with tables that have a d as the second character of the name and to send the output to the db2look.sql file:

```
db2look -d department -e -tw _d% -o db2look.sql
```

- The **db2look** command uses the LIKE predicate when evaluating which table names match the pattern specified by the *Tname* argument. Because the LIKE predicate is used, if either the **\_** character or the **%** character is part of the table name, the backslash (**\**) escape character must be used immediately before the **\_** or the **%**. In this situation, neither the **\_** nor the **%** can be used as a wildcard character in *Tname*. For example, to generate the DDL statements for objects in the DEPARTMENT database associated with tables that have a percent sign in the neither the first nor the last position of the name:

```
db2look -d department -e -tw string\%string
```

- Case-sensitive and multi-word table names must be enclosed by both a backslash and double quotation marks. For example:

```
\ "My Table"
```

- The **-tw** option can be used with the **-x** option (to generate GRANT privileges), the **-m** option (to return table and column statistics), and the **-l** option (to generate the DDL for user-defined table spaces, database partition groups, and buffer pools). If the **-t** option is specified with the **-tw** option, the **-t** option (and its associated *Tname* argument) is ignored.
- The **-tw** option cannot be used to generate the DDL for tables (and their associated objects) that reside on federated data sources, or on DB2 Universal Database for z/OS and OS/390, DB2 Universal Database for iSeries, or DB2 Server for VSE & VM.
- The **-tw** option is only supported via the CLP.

When requesting DDL on systems using the database partitioning feature, a warning message will be displayed in place of the DDL for table spaces that exist on inactive database partitions. To ensure proper DDL is produced for all table spaces all database partitions must be activated.

### Related concepts:

- "Statistics for modeling production databases" in *Performance Guide*

### Related reference:

- "LIKE predicate" in *SQL Reference, Volume 1*

## db2ls - List installed DB2 products and features

Lists the DB2 products and features installed on your system, including the DB2 Version 9 HTML documentation and DB2 Version 8 products. With the ability to install multiple copies of DB2 products on your system and the flexibility to install DB2 products and features in the path of your choice, you can use the **db2ls** command to list:

- where DB2 products are installed on your system and list the DB2 product level.
- all or specific DB2 products and features in a particular installation path.

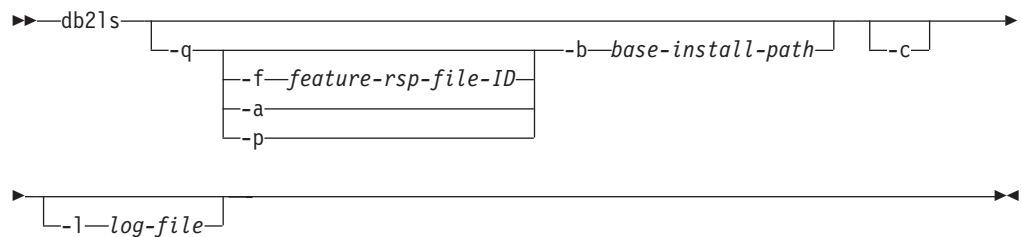
### Authorization:

None.

### Required Connection:

None.

### Command syntax:



### Command parameters:

- q** Signifies that the query is to list installed DB2 products and features. By default, only the visible components (features) are displayed unless the **-a** parameter is also specified.
- f** *feature-rsp-file-ID*  
Queries for the specific feature, if it is installed. If it is not installed, the return code from the program is non-zero, otherwise the return code is zero.
- a** Lists all hidden components as well as visible features. The **db2ls** command only lists visible features by default.
- p** Lists products only. This will give a brief list of which products the customer has installed rather than listing the features.
- b** *base-install-path*  
When using the global **db2ls** command in `/usr/local/bin`, you need to specify which directory you are querying. The global **db2ls** command will simply call the **db2ls** from that install path and pass in the rest of the parameters.
- c** Prints the output as a colon-separated list of entries rather than column-based. This allows you to programmatically with this information. The first line of output will be a colon-separated list of tokens to describe each entry. This first line will start with a hash character ("**#**") to make it easy to ignore programmatically.

## db2ls - List installed DB2 products and features

**-l** *log-file*

Trace logfile to use for debugging purposes.

### Examples:

- To query what DB2 database features are installed to a particular path, issue:  
db2ls -q -b /opt/ibm/ese/v9
- To see all DB2 database features installed to a particular path, issue:  
db2ls -q -a -b /opt/ibm/ese/v9
- To check whether a specific DB2 database feature is installed or not, issue:  
db2ls -q -b /opt/ibm/ese/v9 -f <feature>

### Usage Notes:

- At least one DB2 Version 9 product must already be installed for a symbolic link to the **db2ls** command to be available in /usr/local/bin directory.
- The **db2ls** command is the only method to query a DB2 product. You cannot query DB2 products using Linux or UNIX operating system native utilities such as pkgadd, rpm, SMIT, or swinstall. Any existing scripts containing a native installation utility that you use to interface and query with DB2 installations will need to change.
- You cannot use the **db2ls** command on Windows operating systems.

### Related reference:

- “ENV\_PROD\_INFO administrative view – Retrieve information about installed DB2 products” in *Administrative SQL Routines and Views*
- “db2\_deinstall - Uninstall DB2 products or features ” on page 10

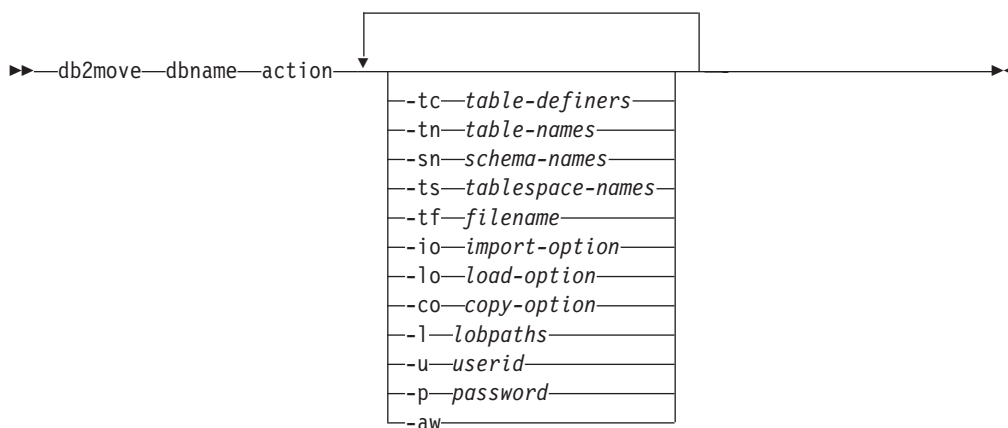
## db2move - Database movement tool

This tool, when used in the EXPORT/IMPORT/LOAD mode, 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. Tables with structured type columns are not moved when this tool is used. When used in the COPY mode, this tool facilitates the duplication of a schema.

### 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:



### Command parameters:

#### dbname

Name of the database.

**action** Must be one of:

#### EXPORT

Exports all tables that meet the filtering criteria in options. If no options are specified, exports all the tables. Internal staging information is stored in the db2move.lst file.

#### IMPORT

Imports all tables listed in the internal staging file db2move.lst. Use the -io option for IMPORT specific actions.

#### LOAD

Loads all tables listed in the internal staging file db2move.lst. Use the -lo option for LOAD specific actions.

**COPY** Duplicates a schema(s) into a target database. Use the -sn option to

## db2move - Database Movement Tool

specify one or more schemas. See the `-co` option for COPY specific options. Use the `-tn` or `-tf` option to filter tables in LOAD\_ONLY mode.

See below for a list of files that are generated during each action.

**-tc** table-definers. The default is all definers.

This is an EXPORT action only. If specified, only those tables created by the definers listed with this option are exported. If not specified, the default is to use all definers. When specifying multiple definers, they must be separated by commas; no blanks are allowed between definer IDs. 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 or COPY action only. If specified, only those tables whose names match exactly those in the specified string are exported or copied. If not specified, the default is to use all user tables. When specifying multiple table names, they must be separated by commas; no blanks are allowed between table names. When using the COPY action, the table names should be listed with their schema qualifier in the format `"schema"."table"`. When using the EXPORT action, the table names should be listed unqualified. This option can be used with the `"-tc"` table-definers option to select the tables for export. **db2move** will only act on those tables whose names match the specified table names and whose definers match the specified table definers.

For export, an asterisk (\*) can be used as a wildcard character that can be placed anywhere in the string.

**-sn** schema-names. The default for EXPORT is all schemas (not for COPY).

If specified, only those tables whose schema names match exactly will be exported or copied. If multiple schema names are specified, they must be separated by commas; no blanks are allowed between schema names. Schema names of less than 8 character are padded to 8 characters in length.

In the case of export:

If the asterisk wildcard character (\*) is used in the schema names, it will be changed to a percent sign (%) and the table name (with percent sign) will be used in the LIKE predicate of the WHERE clause. If not specified, the default is to use all schemas. If used with the `-tn` or `-tc` option, **db2move** will only act on those tables whose schemas match the specified schema names and whose definers match the specified definers. A schema name 'fred' has to be specified `"-sn fr*d*"` instead of `"-sn fr*d"` when using an asterisk.

**-ts** tablespace-names. The default is all table spaces.

This is an EXPORT action only. If this option is specified, only those tables that reside in the specified table space will be exported. If the asterisk wildcard character (\*) is used in the table space name, it will be changed to a percent sign (%) and the table name (with percent sign) will be used in the LIKE predicate in the WHERE clause. If the `-ts` option is not specified,



the default is to use all table spaces. If multiple table space names are specified, they must be separated by commas; no blanks are allowed between table space names. Table space names less than 8 characters are padded to 8 characters in length. For example, a table space name 'mytb' has to be specified "-ts my\*b\*" instead of "-sn my\*b" when using the asterisk.

**-tf** filename

This is an EXPORT or COPY action only. If specified, only the tables listed in the given file will be exported or copied. The tables should be listed one per line, and each table should be fully qualified. Here is an example of the contents of a file:

```
"SCHEMA1"."TABLE NAME1"
"SCHEMA NAME77"."TABLE155"
```

**-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.

**-co** When the **db2move** action is COPY, the following -co follow-on options will be available:

**"TARGET\_DB <db name> [USER <userid> USING <password>]"**

Allows the user to specify the name of the target database and the user/password. (The source database user/password can be specified using the existing -p and -u options). The USER/USING clause is optional. If USER specifies a userid, then the password must either be supplied following the USING clause, or if it's not specified, then **db2move** will prompt for the password information. The reason for prompting is for security reasons discussed below. TARGET\_DB is a mandatory option for the COPY action. The TARGET\_DB cannot be the same as the source database. The ADMIN\_COPY\_SCHEMA procedure can be used for copying schemas within the same database. The COPY action requires inputting at least one schema (-sn) or one table (-tn or -tf).

Running multiple **db2move** commands to copy schemas from one database to another will result in deadlocks. Only one **db2move** command should be issued at a time. Changes to tables in the source schema during copy processing may mean that the data in the target schema is not identical following a copy.

**"MODE"**

**DDL\_AND\_LOAD**

Creates all supported objects from the source schema, and populates the tables with the source table data. This is the default option.

**DDL\_ONLY**

Creates all supported objects from the source schema, but does not repopulate the tables.

**LOAD\_ONLY**

Loads all specified tables from the source database to the target database. The tables must already exist on the target.

This is an optional option that is only used with the COPY action.

### “SCHEMA\_MAP”

Allows user to rename schema when copying to target. Provides a list of the source-target schema mapping, separated by commas, surrounded by brackets. e.g schema\_map ((s1, t1), (s2, t2)). This would mean objects from schema s1 will be copied to schema t1 on the target; objects from schema s2 will be copied to schema t2 on the target. The default, and recommended, target schema name is the source schema name. The reason for this is **db2move** will not attempt to modify the schema for any qualified objects within object bodies. Therefore, using a different target schema name may lead to problems if there are qualified objects within the object body.

For example: create view F00.v1 as 'select c1 from F00.t1'

In this case, copy of schema FOO to BAR, v1 will be regenerated as: create view BAR.v1 as 'select c1 from F00.t1'

This will either fail since schema FOO does not exist on the target database, or have an unexpected result due to FOO being different than BAR. Maintaining the same schema name as the source will avoid these issues. If there are cross dependencies between schemas, all inter-dependant schemas must be copied or there may be errors copying the objects with the cross dependencies.

For example: create view F00.v1 as 'select c1 from BAR.t1'

In this case, the copy of v1 will either fail if BAR is not copied as well, or have an unexpected result if BAR on the target is different than BAR from the source. **db2move** will not attempt to detect cross schema dependencies.

This is an optional option that is only used with the COPY action.

### “NONRECOVERABLE”

This option allows the user to override the default behaviour of the load to be done with COPY-NO. With the default behaviour, the user will be forced to take backups of each tablespace that was loaded into. When specifying this NONRECOVERABLE keyword, the user will not be forced to take backups of the tablespaces immediately. It is, however, highly recommended that the backups be taken as soon as possible to ensure the newly created tables will be properly recoverable. This is an optional option available to the COPY action.

### “OWNER”

Allows the user to change the owner of each new object created in the target schema after a successful COPY. The default owner of the target objects will be the connect user; if this option is specified, ownership will be transferred to the new owner. This option is pending due to containability Q1/2006 delivery but this parameter will be in the first design. This is an optional option available to the COPY action.

### “TABLESPACE\_MAP”

The user may specify tablespace name mappings to be used instead of the tablespaces from the source system during a copy. This will be an array of tablespace mappings surrounded by brackets. For example, tablespace\_map ((TS1,

TS2), (TS3, TS4)). This would mean that all objects from tablespace TS1 will be copied into tablespace TS2 on the target database and objects from tablespace TS3 will be copied into tablespace TS4 on the target. In the case of ((T1, T2), (T2, T3)), all objects found in T1 on the source database will be recreated in T2 on the target database and any objects found in T2 on the source database will be recreated in T3 on the target database. The default is to use the same tablespace name as from the source, in which case, the input mapping for this tablespace is not necessary. If the specified tablespace does not exist, the copy of the objects using that tablespace will fail and be logged in the error file.

The user also has the option of using the SYS\_ANY keyword to indicate that the target tablespace should be chosen using the default tablespace selection algorithm. In this case, **db2move** will be able to choose any available tablespace to be used as the target. The SYS\_ANY keyword can be used for all tablespaces, example: `tablespace_map SYS_ANY`. In addition, the user can specify specific mappings for some tablespaces, and the default tablespace selection algorithm for the remaining. For example, `tablespace_map ((TS1, TS2), (TS3, TS4), SYS_ANY)`. This indicates that tablespace TS1 is mapped to TS2, TS3 is mapped to TS4, but the remaining tablespaces will be using a default tablespace target. The SYS\_ANY keyword is being used since it's not possible to have a tablespace starting with "SYS".

This is an optional option available to the COPY action.

- l lobpaths. For IMPORT and EXPORT, if this option is specified, it will be also used for XML paths. The default is the current directory.  
  
This option specifies the absolute path names where LOB or XML files are created (as part of EXPORT) or searched for (as part of IMPORT or LOAD). When specifying multiple paths, each must be separated by commas; no blanks are allowed between paths. If multiple paths are specified, EXPORT will use them in round-robin fashion. It will write one LOB document to the first path, one to the second path, and so on up to the last, then back to the first path. The same is true for XML documents. If files are not found in the first path (during IMPORT or LOAD), the second path will be used, and so on.
- 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. When the -p option is specified, but the password not supplied, **db2move** will prompt for the password. This is done for security reasons. Inputting the password through command line creates security issues. For example, a `ps -ef` command would display the password. If, however, **db2move** is invoked through a script, then the passwords will have to be supplied. If the command is issued on a client connecting to a remote server, user ID and password should be specified.
- aw Allow Warnings. When '-aw' is not specified, tables that experience warnings during export are not included in the `db2move.lst` file (although that table's `.ixf` file and `.msg` file are still generated). In some scenarios

## db2move - Database Movement Tool

(such as data truncation) the user might wish to allow such tables to be included in the db2move.lst file. Specifying this option allows tables which receive warnings during export to be included in the .lst file.

### Examples:

- To export all tables in the SAMPLE database (using default values for all options), issue:  

```
db2move sample export
```
- To export all tables created by userid1 or user IDs LIKE us\*rid2, and with the name tname1 or table names LIKE %tname2, issue:  

```
db2move sample export -tc userid1,us*rid2 -tn tname1,*tname2
```
- To import all tables in the SAMPLE database (LOB paths D:\LOBPATH1 and C:\LOBPATH2 are to be searched for LOB files; this example is applicable to Windows operating systems only), issue:  

```
db2move sample import -l D:\LOBPATH1,C:\LOBPATH2
```
- To load all tables in the SAMPLE database (/home/userid/lobpath subdirectory and the tmp subdirectory are to be searched for LOB files; this example is applicable to Linux and UNIX-based systems only), issue:  

```
db2move sample load -l /home/userid/lobpath,/tmp
```
- To import all tables in the SAMPLE database in REPLACE mode using the specified user ID and password, issue:  

```
db2move sample import -io replace -u userid -p password
```
- To duplicate schema schema1 from source database dbsrc to target database dbtgt, issue:  

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
```
- To duplicate schema schema1 from source database dbsrc to target database dbtgt, rename the schema to newschema1 on the target, and map source tablespace ts1 to ts2 on the target, issue:  

```
db2move dbsrc COPY -sn schema1 -co TARGET_DB dbtgt USER myuser1 USING mypass1
SCHEMA_MAP ((schema1,newschema1)) TABLESPACE_MAP ((ts1,ts2), SYS_ANY))
```

### Usage notes:

- Loading data into tables containing XML columns is not supported. The workaround is to manually issue the **IMPORT** or **EXPORT** commands, or use the **db2move -Export** and **db2move -Import** behaviour. If these tables also contain generated always identity columns, data cannot be imported into the tables.
- 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. If the import utility with the REPLACE\_CREATE option is used to create the tables on the target database, then the limitations outlined in Using import to recreate an exported table are imposed. If unexpected errors are encountered during the **db2move** import phase when the REPLACE\_CREATE option is used, examine the appropriate tabnnn.msg message file and consider that the errors might be the result of the limitations on table creation.
- When export, import, or load APIs are called by **db2move**, the FileTypeMod parameter is set to lobsinfile. That is, LOB data is kept in file separate from the PC/IXF file, for every table.
- The LOAD command must be run locally on the machine where the database and the data file reside. When the load API is called by **db2move**, the

NONRECOVERABLE option is used. If logretain is on, and the -lo option is INSERT, the load operation marks the table as inaccessible and it must be dropped. 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. Performance for the **DB2MOVE** command with the IMPORT action can be improved by altering the default buffer pool, IBMDEFAULTBP; and by updating the configuration parameters sortheap, util\_heap\_sz, logfilsz, and logprimary.

- For more information on the NONRECOVERABLE recoverability option see the Data Movement Utilities Guide and Reference.

### Files Required/Generated When Using EXPORT:

- Input: None.

- Output:

|                    |                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>EXPORT.out</b>  | The summarized result of the EXPORT action.                                                                                                                                                                                                                                                                                                                                     |
| <b>db2move.lst</b> | 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 <b>db2move</b> IMPORT or LOAD action.                                                                                                                     |
| <b>tabnnn.ixf</b>  | The exported PC/IXF file of a specific table.                                                                                                                                                                                                                                                                                                                                   |
| <b>tabnnn.msg</b>  | The export message file of the corresponding table.                                                                                                                                                                                                                                                                                                                             |
| <b>tabnnnc.yyy</b> | 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. |
| <b>system.msg</b>  | 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:

|                    |                                        |
|--------------------|----------------------------------------|
| <b>db2move.lst</b> | An output file from the EXPORT action. |
| <b>tabnnn.ixf</b>  | An output file from the EXPORT action. |
| <b>tabnnnc.yyy</b> | An output file from the EXPORT action. |

- Output:

|                   |                                                     |
|-------------------|-----------------------------------------------------|
| <b>IMPORT.out</b> | The summarized result of the IMPORT action.         |
| <b>tabnnn.msg</b> | The import message file of the corresponding table. |

### Files Required/Generated When Using LOAD:

- Input:

|                    |                                        |
|--------------------|----------------------------------------|
| <b>db2move.lst</b> | An output file from the EXPORT action. |
| <b>tabnnn.ixf</b>  | An output file from the EXPORT action. |
| <b>tabnnnc.yyy</b> | An output file from the EXPORT action. |

## db2move - Database Movement Tool

- Output:
  - LOAD.out**      The summarized result of the LOAD action.
  - tabnnn.msg**    The LOAD message file of the corresponding table.

### Files Required/Generated When Using COPY:

- Input: None
- Output:
  - COPYSCHEMA.msg**  
An output file from the COPY action.
  - COPYSCHEMA.err**  
An output file from the COPY action.
  - LOADTABLE.err**  
An output file from the COPY action.
  - LOADTABLE.msg**  
An output file from the COPY action.

These files are timestamped and all files that are generated from one run will have the same timestamp.

### Related reference:

- “db2look - DB2 statistics and DDL extraction tool ” on page 146

## db2mqdsn - MQ listener

Invokes the asynchronous MQListener to monitor a set of WebSphere MQ message queues, passing messages that arrive on them to configured DB2 stored procedures. It can also perform associated administrative and configuration tasks. MQListener configuration information is stored in a DB2 database and consists of a set of named configurations, including a default. Each configuration is composed of a set of tasks. MQListener tasks are defined by the message queue from which to retrieve messages and the stored procedure to which they will be passed. The message queue description must include the name of the message queue and its queue manager, if it is not the default. Information about the stored procedure must include the database in which it is defined, a user name and password with which to access the database, and the procedure name and schema.

On Linux and UNIX operating systems, this utility is located in the DB2DIR/instance directory, where DB2DIR is the location where the current version of the DB2 database product is installed.

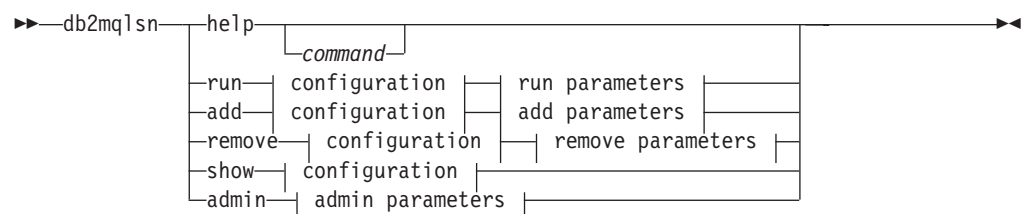
On Windows operating systems, this utility is located in the DB2PATH\sql1lib\bin directory, where DB2PATH is the location where the current version of the DB2 database product is installed.

For more information about controlling access to WebSphere MQ objects, refer to the *WebSphere MQ System Administration Guide* (SC34-6068-00).

### Authorization:

- All options except **db2mqdsn admin** access the MQListener configuration in the configDB database. The connection is made as configUser or, if no user is specified, an implicit connection is attempted. The user in whose name the connection is made must have EXECUTE privilege on package mqlConfi.
- To access MQ objects with the **db2mqdsn run** and **db2mqdsn admin** options, the user who executes the program must be able to open the appropriate MQ objects.
- To execute the **db2mqdsn run** option successfully, the dbUser specified in the **db2mqdsn add** option that created the task must have EXECUTE privilege on the specified stored procedure, and must have EXECUTE privilege on the package mqlRun in the dbName database.

### Command syntax:



### configuration:

```

 |--configDB--configuration database name-->>

```

## db2mqIsn - MQ Listener

```
└─configUser—user ID—configPwd—password┘
└─config—configuration name┘
```

### run parameters:

```
└─adminQueue—admin queue name┘
└─adminQMgr—admin queue manager┘
```

### add parameters:

```
└─inputQueue—input queue name┘
└─queueManager—queue manager name┘
└─procSchema—stored procedure schema—procName—stored procedure name┘
└─dbName—stored procedure database┘
└─dbUser—user ID—dbPwd—password┘
└─mqCoordinated┘
└─numInstances—number of instances to run┘
```

### remove parameters:

```
└─inputQueue—input queue name┘
└─queueManager—queue manager name┘
```

### admin parameters:

```
└─adminQueue—admin queue name┘
└─adminQueueList—namelist of admin queue names┘
└─adminQMgr—admin queue manager┘
└─adminCommand—shutdown┘
└─restart┘
```

### Command parameters:

#### help *command*

Supplies detailed information about a particular command. If you do not give a command name, then a general help message is displayed.

#### —configDB *configuration database*

Name of the database that contains the configuration information.

#### —configUser *user ID* —configPwd *password*

Authorization information with which to access the configuration database.

#### —config *configuration name*

You can group individual tasks into a configuration. By doing this you can run a group of tasks together. If you do not specify a configuration name, then the utility runs the default configuration.

#### run

#### —adminQueue *admin queue name* —adminQMgr *admin queue manager*

This is the queue on which the MQListener listens for administration commands. If you do not specify a queue manager,



then the utility uses the configured default queue manager. If you do not specify an `adminQueue`, then the application does not receive any administration commands (such as shut down or restart) through the message queue.

#### add

- inputQueue** *input queue name* **-queueManager** *queue manager name*  
This is the queue on which the MQListener listens for messages for this task. If you do not specify a queue manager, the utility uses the default queue manager configured in WebSphere MQ.
- procSchema** *stored procedure schema* **-procName** *stored procedure name*  
The stored procedure to which MQListener passes the message when it arrives.
- dbName** *stored procedure database*  
MQListener passes the message to a stored procedure. This is the database in which the stored procedure is defined.
- dbUser** *user ID* **-dbPwd** *password*  
The user on whose behalf the stored procedure is invoked.
- mqCoordinated**  
This indicates that reading and writing to the WebSphere MQ message queue should be integrated into a transaction together with the DB2 stored procedure call. The entire transaction is coordinated by the WebSphere MQ coordinator. (The queue manager must also be configured to coordinate a transaction in this way. See the WebSphere MQ documentation for more information.) By default, the message queue operations are not part of the transaction in which the stored procedure is invoked.
- numInstances** *number of instances to run*  
The number of duplicate instances of this task to run in this configuration. If you do not specify a value, then only one instance is run.

#### remove

- inputQueue** *input queue name* **-queueManager** *queue manager name*  
This is the queue and queue manager that define the task that will be removed from the configuration. The combination of input queue and queue manager is unique within a configuration.

#### admin

- adminQueue** *admin queue name* **-adminQueueList** *namelist of admin queue names* **-adminQMGr** *admin queue manager*  
The queue or namelist of queue names on which to send the admin command. If you do not specify a queue manager, the utility uses the default queue manager that is configured in WebSphere MQ.
- adminCommand** *admin command*  
Submits a command. The command can be either shutdown or restart. Shutdown causes a running MQListener to exit when the listener finishes processing the current message. Restart performs a shutdown, and then reads the configuration again and restarts.

#### Examples:

```
db2mq1sn show -configDB sampleDB -config nightlies
```

## db2mq1sn - MQ Listener

```
db2mq1sn add -configDB sampleDB -config nightlies -inputQueue app3
-procSchema imauser -procName proc3 -dbName aDB -dbUser imauser -dbPwd aSecret
db2mq1sn run -configDB -config nightlies
```

### Related reference:

- “disable\_MQFunctions” on page 299
- “enable\_MQFunctions” on page 301

## db2mscs - Set up Windows failover utility

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

### Authorization:

The user must be logged on to a domain user account that belongs to the Administrators group of each machine in the MSCS cluster.

### Command syntax:

```

>> db2mscs [-f:input_file] [-u:instance_name]

```

### Command parameters:

#### -f:input\_file

Specifies the DB2MSCS.CFG input file to be used by the MSCS utility. If this parameter is not specified, the DB2MSCS utility reads the DB2MSCS.CFG file that is in the current directory.

#### -u:instance\_name

This option allows you to undo the db2mscs operation and revert the instance back to the non-MSCS instance specified by instance\_name.

### Usage notes:

The DB2MSCS utility is a standalone command line utility used to transform a non-MSCS instance into an MSCS instance. The utility will create all MSCS groups, resources, and resource dependencies. It will also copy all DB2 information stored in the Windows registry to the cluster portion of the registry as well as moving the instance directory to a shared cluster disk. The DB2MSCS utility takes as input a configuration file provided by the user specifying how the cluster should be set up. The DB2MSCS.CFG file is an ASCII text file that contains parameters that are read by the DB2MSCS utility. You specify each input parameter on a separate line using the following format: PARAMETER\_KEYWORD=parameter\_value. For example:

```

CLUSTER_NAME=FINANCE
GROUP_NAME=DB2 Group
IP_ADDRESS=9.21.22.89

```

Two example configuration files can be found in the CFG subdirectory under the DB2 install directory. The first, DB2MSCS.EE, is an example for single-partition database environments. The second, DB2MSCS.EEE, is an example for partitioned database environments.

The parameters for the DB2MSCS.CFG file are as follows:

#### DB2\_INSTANCE

The name of the DB2 instance. This parameter has a global scope and should be specified only once in the DB2MSCS.CFG file.

#### DAS\_INSTANCE

The name of the DB2 Admin Server instance. Specify this parameter to

## db2mscs - Set up Windows Failover Utility

migrate the DB2 Admin Server to run in the MSCS environment. This parameter has a global scope and should be specified only once in the DB2MSCS.CFG file.

### CLUSTER\_NAME

The name of the MSCS cluster. All the resources specified following this line are created in this cluster until another CLUSTER\_NAME parameter is specified.

### DB2\_LOGON\_USERNAME

The user name of the domain account for the DB2 service (specified as *domain\user*). This parameter has a global scope and should be specified only once in the DB2MSCS.CFG file.

### DB2\_LOGON\_PASSWORD

The password of the domain account for the DB2 service. This parameter has a global scope and should be specified only once in the DB2MSCS.CFG file.

### GROUP\_NAME

The name of the MSCS group. If this parameter is specified, a new MSCS group is created if it does not exist. If the group already exists, it is used as the target group. Any MSCS resource specified after this parameter is created in this group or moved into this group until another GROUP\_NAME parameter is specified. Specify this parameter once for each group.

### DB2\_NODE

The database partition number of the database partition server (or database partition) to be included in the current MSCS group. If multiple logical database partitions exist on the same machine, each database partition requires a separate DB2\_NODE parameter. Specify this parameter after the GROUP\_NAME parameter so that the DB2 resources are created in the correct MSCS group. This parameter is required for a multi-partitioned database system.

### IP\_NAME

The name of the IP Address resource. The value for the IP\_NAME is arbitrary, but it must be unique in the cluster. When this parameter is specified, an MSCS resource of type IP Address is created. This parameter is required for remote TCP/IP connections. This parameter is optional in a single partition database environment. A recommended name is the hostname that corresponds to the IP address.

### IP\_ADDRESS

The TCP/IP address for the IP resource specified by the preceding IP\_NAME parameter. This parameter is required if the IP\_NAME parameter is specified. This is a new IP address that is not used by any machine in the network.

### IP\_SUBNET

The TCP/IP subnet mask for the IP resource specified by the preceding IP\_NAME parameter. This parameter is required if the IP\_NAME parameter is specified.

### IP\_NETWORK

The name of the MSCS network to which the preceding IP Address resource belongs. This parameter is optional. If it is not specified, the first MSCS network detected by the system is used. The name of the MSCS

network must be entered exactly as seen under the Networks branch in Cluster Administrator. The previous four IP keywords are used to create an IP Address resource.

### NETNAME\_NAME

The name of the Network Name resource. Specify this parameter to create the Network Name resource. This parameter is optional for single partition database environment. You must specify this parameter for the instance owning machine in a partitioned database environment.

### NETNAME\_VALUE

The value for the Network Name resource. This parameter must be specified if the NETNAME\_NAME parameter is specified.

### NETNAME\_DEPENDENCY

The name for the IP resource that the Network Name resource depends on. Each Network Name resource must have a dependency on an IP Address resource. This parameter is optional. If it is not specified, the Network Name resource has a dependency on the first IP resource in the group.

### SERVICE\_DISPLAY\_NAME

The display name of the Generic Service resource. Specify this parameter if you want to create a Generic Service resource.

### SERVICE\_NAME

The service name of the Generic Service resource. This parameter must be specified if the SERVICE\_DISPLAY\_NAME parameter is specified.

### SERVICE\_STARTUP

Optional startup parameter for the Generic Resource service.

### DISK\_NAME

The name of the physical disk resource to be moved to the current group. Specify as many disk resources as you need. The disk resources must already exist. When the DB2MSCS utility configures the DB2 instance for failover support, the instance directory is copied to the first MSCS disk in the group. To specify a different MSCS disk for the instance directory, use the INSTPROF\_DISK parameter. The disk name used should be entered exactly as seen in Cluster Administrator.

### INSTPROF\_DISK

An optional parameter to specify an MSCS disk to contain the DB2 instance directory. If this parameter is not specified the DB2MSCS utility uses the first disk that belongs to the same group.

### INSTPROF\_PATH

An optional parameter to specify the exact path where the instance directory will be copied. This parameter *must* be specified when using IPSHADisks, a ServerRAID Netfinity disk resource (for example, INSTPROF\_PATH=p:\db2profs). INSTPROF\_PATH will take precedence over INSTPROF\_DISK if both are specified.

### TARGET\_DRVMAP\_DISK

An optional parameter to specify the target MSCS disk for database drive mapping for a the multi-partitioned database system. This parameter will specify the disk the database will be created on by mapping it from the drive the create database command specifies. If this parameter is not specified, the database drive mapping must be manually registered using the DB2DRVMP utility.

## db2mscs - Set up Windows Failover Utility

### DB2\_FALLBACK

An optional parameter to control whether or not the applications should be forced off when the DB2 resource is brought offline. If not specified, then the setting for DB2\_FALLBACK will be YES. If you do not want the applications to be forced off, then set DB2\_FALLBACK to NO.

### Related reference:

- “db2drvmp - DB2 database drive map ” on page 88

## db2mtrk - Memory tracker

Provide complete report of memory status, for instances, databases and agents. This command outputs the following memory pool allocation information:

- Current size
- Maximum size (hard limit)
- Largest size (high water mark)
- Type (identifier indicating function for which memory will be used)
- Agent who allocated pool (only if the pool is private)

The same information is also available from the Snapshot monitor.

### Scope

In a partitioned database environment, this command can be invoked from any database partition defined in the db2nodes.cfg file. It returns information only for that database partition. This command does not return information for remote servers.

### Authorization:

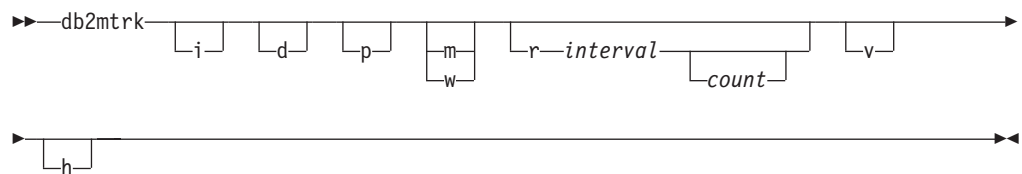
One of the following:

- sysadm
- sysctrl
- sysmaint

### Required Connection:

Instance. The application creates a default instance attachment if one is not present.

### Command Syntax:



### Command Parameters:

- i** Show instance level memory.
- d** Show database level memory.
- p** Show private memory.
- m** Show maximum values for each pool.
- w** Show high watermark values for each pool.
- r** Repeat mode

*interval*

Number of seconds to wait between subsequent calls to the memory tracker (in repeat mode).

*count* Number of times to repeat.

## db2mtrk - Memory Tracker

- v** Verbose output.
- h** Show help screen. If you specify *-h*, only the help screen appears. No other information is displayed.

### Examples:

The following call returns database and instance normal values and repeats every 10 seconds:

```
db2mtrk -i -d -v -r 10
```

Consider the following output samples:

The command `db2mtrk -i -d -p` displays the following output:

Tracking Memory on: 2006/01/17 at 15:24:38

Memory for instance

```
monh other
576.0K 8.0M
```

Memory for database: AJSTORM

```
utilh pckcacheh catcacheh bph (1) bph (S32K) bph (S16K) bph (S8K)
64.0K 640.0K 128.0K 34.2M 576.0K 320.0K 192.0K

bph (S4K) shsorth lockh dbh other
128.0K 64.0K 9.6M 4.8M 192.0K
```

Memory for database: CMGARCIA

```
utilh pckcacheh catcacheh bph (1) bph (S32K) bph (S16K) bph (S8K)
64.0K 640.0K 128.0K 34.2M 576.0K 320.0K 192.0K

bph (S4K) shsorth lockh dbh other
128.0K 64.0K 9.6M 4.8M 192.0K
```

Memory for agent 970830

```
other apph appctlh
64.0K 64.0K 64.0K
```

Memory for agent 4460644

```
other appctlh apph
64.0K 64.0K 64.0K
```

The command `db2mtrk -i -d -p -v` displays the following output:

Tracking Memory on: 2006/01/17 at 15:25:36

Memory for instance

```
Database Monitor Heap is of size 589824 bytes
Other Memory is of size 8388608 bytes
Total: 8978432 bytes
```

Memory for database: AJSTORM

```
Backup/Restore/Util Heap is of size 65536 bytes
Package Cache is of size 655360 bytes
Catalog Cache Heap is of size 131072 bytes
Buffer Pool Heap (1) is of size 35913728 bytes
Buffer Pool Heap (System 32k buffer pool) is of size 589824 bytes
Buffer Pool Heap (System 16k buffer pool) is of size 327680 bytes
Buffer Pool Heap (System 8k buffer pool) is of size 196608 bytes
Buffer Pool Heap (System 4k buffer pool) is of size 131072 bytes
```



Shared Sort Heap is of size 65536 bytes  
Lock Manager Heap is of size 10092544 bytes  
Database Heap is of size 4980736 bytes  
Other Memory is of size 196608 bytes  
Total: 53346304 bytes

**Memory for database: CMGARCIA**

Backup/Restore/Util Heap is of size 65536 bytes  
Package Cache is of size 655360 bytes  
Catalog Cache Heap is of size 131072 bytes  
Buffer Pool Heap (1) is of size 35913728 bytes  
Buffer Pool Heap (System 32k buffer pool) is of size 589824 bytes  
Buffer Pool Heap (System 16k buffer pool) is of size 327680 bytes  
Buffer Pool Heap (System 8k buffer pool) is of size 196608 bytes  
Buffer Pool Heap (System 4k buffer pool) is of size 131072 bytes  
Shared Sort Heap is of size 65536 bytes  
Lock Manager Heap is of size 10092544 bytes  
Database Heap is of size 4980736 bytes  
Other Memory is of size 196608 bytes  
Total: 53346304 bytes

**Memory for agent 970830**

Other Memory is of size 65536 bytes  
Application Heap is of size 65536 bytes  
Application Control Heap is of size 65536 bytes  
Total: 196608 bytes

**Memory for agent 4460644**

Other Memory is of size 65536 bytes  
Application Control Heap is of size 65536 bytes  
Application Heap is of size 65536 bytes  
Total: 196608 bytes

**Usage Notes:****Notes:**

1. When no flags are specified, usage is returned.
2. One of the -d, -h, -i, or -p flag must be specified.
3. When the -p flag is specified, detailed private memory usage information is returned ordered by agent ID.
4. The "Other Memory" reported is the memory associated with the overhead of operating the database management system.
5. In some cases (such as the package cache) the maximum size displayed will be larger than the value assigned to the configuration parameter. In such cases, the value assigned to the configuration parameter is used as a 'soft limit', and the pool's actual memory usage might grow beyond the configured size.
6. For the buffer pool heaps, the number specified in the parentheses is either the buffer pool ID, or indicates that this buffer pool is one of the system buffer pools.
7. The maximum size that the memory tracker reports for some heaps is the amount of physical memory on the machine. These heaps are called unbounded heaps and are declared with an unlimited maximum size because when the heaps are declared, it is not clear how much memory they will require at peak times. Although these heaps are not strictly bounded by the physical memory on the machine, they are reported as the maximum size because it is a reasonable approximation.

**Related concepts:**

- "Memory allocation in DB2" in *Performance Guide*

### db2nchg - Change database partition server configuration

Modifies database partition server configuration. This includes moving the database partition server (node) from one machine to another; changing the TCP/IP host name of the machine; and selecting a different logical port number or a different network name for the database partition server (node). This command can only be used if the database partition server is stopped.

This command is available on Windows-based operating systems only.

#### Authorization:

Local Administrator

#### Command syntax:

```
db2nchg /n:dbpartitionnum [/i:instance_name] [/u:user,password] [/p:logical_port] [/h:hostname] [/m:machine_name] [/g:network_name]
```

#### Command parameters:

##### **/n:dbpartitionnum**

Specifies the database partition number of the database partition server's configuration that is to be changed.

##### **/i:instance\_name**

Specifies the instance in which this database partition server participates. If a parameter is not specified, the default is the current instance.

##### **/u:username,password**

Specifies the user name and password. If a parameter is not specified, the existing user name and password will apply.

##### **/p:logical\_port**

Specifies the logical port for the database partition server. This parameter must be specified to move the database partition server to a different machine. If a parameter is not specified, the logical port number will remain unchanged.

##### **/h:host\_name**

Specifies TCP/IP host name used by FCM for internal communications. If this parameter is not specified, the host name will remain the same.

##### **/m:machine\_name**

Specifies the machine where the database partition server will reside. The database partition server can only be moved if there are no existing databases in the instance.

##### **/g:network\_name**

Changes the network name for the database partition server. This parameter can be used to apply a specific IP address to the database partition server when there are multiple IP addresses on a machine. The network name or the IP address can be entered.

## db2nchg - Change Database Partition Server Configuration

### Examples:

To change the logical port assigned to database partition 2, which participates in the instance TESTMPP, to logical port 3, enter the following command:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

### Related reference:

- “db2ncrt - Add database partition server to an instance ” on page 178
- “db2ndrop - Drop database partition server from an instance ” on page 180

---

### db2ncrt - Add database partition server to an instance

Adds a database partition server (node) to an instance.

This command is available on Windows operating systems only.

#### Scope:

If a database partition server is added to a computer where an instance already exists, a database partition server is added as a logical database partition server to the computer. If a database partition server is added to a computer where an instance does not exist, the instance is added and the computer becomes a new physical database partition server. This command should not be used if there are databases in an instance. Instead, the START DATABASE MANAGER command should be issued with the ADD DBPARTITIONNUM option. This ensures that the database is correctly added to the new database partition server. It is also possible to add a database partition server to an instance in which a database has been created. The db2nodes.cfg file should not be edited since changing the file might cause inconsistencies in the partitioned database system.

#### Authorization:

Local Administrator authority on the computer where the new database partition server is added.

#### Command syntax:

```
db2ncrt -/n:—dbpartitionnum—/u:—username,password—
 [/i:—instance_name—] [/m:—machine_name—] [/p:—logical_port—]
 [/h:—host_name—] [/g:—network_name—] [/o:—instance_owning_machine—]
```

#### Command parameters:

##### /n:dbpartitionnum

A unique database partition number which identifies the database partition server. The number entered can range from 1 to 999.

##### /u:domain\_name\username,password

Specifies the domain, logon account name and password for DB2.

##### /i:instance\_name

Specifies the instance name. If a parameter is not specified, the default is the current instance.

##### /m:machine\_name

Specifies the computer name of the Windows workstation on which the database partition server resides. This parameter is required if a database partition server is added on a remote computer.

##### /p:logical\_port

Specifies the logical port number used for the database partition server. If this parameter is not specified, the logical port number assigned will be 0.

## db2ncrt - Add Database Partition Server to an Instance

When creating a logical database partition server, this parameter must be specified and a logical port number that is not in use must be selected. Note the following restrictions:

- Every computer must have a database partition server that has a logical port 0.
- The port number cannot exceed the port range reserved for FCM communications in the `x:\winnt\system32\drivers\etc\` directory. For example, if a range of 4 ports is reserved for the current instance, then the maximum port number is 3. Port 0 is used for the default logical database partition server.

### **/h:host\_name**

Specifies the TCP/IP host name that is used by FCM for internal communications. This parameter is required when the database partition server is being added on a remote computer.

### **/g:network\_name**

Specifies the network name for the database partition server. If a parameter is not specified, the first IP address detected on the system will be used. This parameter can be used to apply a specific IP address to the database partition server when there are multiple IP addresses on a computer. The network name or the IP address can be entered.

### **/o:instance\_owning\_machine**

Specifies the computer name of the instance-owning computer. The default is the local computer. This parameter is required when the **db2ncrt** command is invoked on any computer that is not the instance-owning computer.

### **Examples:**

To add a new database partition server to the instance TESTMPP on the instance-owning computer SHAYER, where the new database partition server is known as database partition 2 and uses logical port 1, enter the following command:

```
db2ncrt /n:2 /u:QBPAULZ\paulz,g1reeky /i:TESTMPP /m:TEST /p:1 /o:SHAYER
```

### **Related reference:**

- “db2nchg - Change database partition server configuration ” on page 176
- “db2ndrop - Drop database partition server from an instance ” on page 180

---

### db2ndrop - Drop database partition server from an instance

Drops a database partition server (node) from an instance that has no databases. If a database partition server is dropped, its database partition number can be reused for a new database partition server. This command can only be used if the database partition server is stopped.

This command is available on Windows-based operating systems only.

#### Authorization:

Local Administrator authority on the machine where the database partition server is being dropped.

#### Command syntax:

```
db2ndrop /n:dbpartitionnum [/i:instance_name]
```

#### Command parameters:

##### /n:dbpartitionnum

A unique database partition number which identifies the database partition server.

##### /i:instance\_name

Specifies the instance name. If a parameter is not specified, the default is the current instance.

#### Examples:

```
db2ndrop /n:2 /i:KMASCI
```

#### Usage notes:

If the instance-owning database partition server (dbpartitionnum 0) is dropped from the instance, the instance becomes unusable. To drop the instance, use the **db2idrop** command.

This command should not be used if there are databases in this instance. Instead, the **db2stop drop nodenum** command should be used. This ensures that the database partition server is correctly removed from the partition database system. It is also possible to drop a database partition server in an instance where a database exists. The **db2nodes.cfg** file should not be edited since changing the file might cause inconsistencies in the partitioned database system.

To drop a database partition server that is assigned to the logical port 0 from a machine that is running multiple logical database partition servers, all other database partition servers assigned to the other logical ports must be dropped first. Each database partition server must have a database partition server assigned to logical port 0.

#### Related reference:

- “db2nchg - Change database partition server configuration ” on page 176
- “db2ncrt - Add database partition server to an instance ” on page 178

## db2osconf - Utility for kernel parameter values

Makes recommendations for kernel parameter values based on the size of a system. The recommended values are high enough for a given system that they can accommodate most reasonable workloads. This command is currently available only for DB2 on HP-UX on 64-bit instances and the Solaris operating system.

### Authorization:

- On DB2 for HP-UX, no authorization is required. To make the changes recommended by the **db2osconf** utility, you must have root access.
- On DB2 for the Solaris operating system, you must have root access or be a member of the sys group.

### Command syntax:

To get the list of currently supported options, enter `db2osconf -h`:

```
db2osconf -h
Usage:
-c # Client only
-f # Compare to current
-h # Help screen
-l # List current
-m <mem in GB> # Specify memory in GB
-n <num CPUs> # Specify number of CPUs
-p <perf level> # Msg Q performance level (0-3)
-s <scale factor> # Scale factor (1-3)
-t <threads> # Number of threads
```

### Command parameters:

- c** The `'-c'` switch is for client only installations. This option is available only on DB2 for the Solaris operating system.
- f** The `'-f'` switch can be used to compare the current kernel parameters with the values that would be recommended by the **db2osconf** utility. The `-f` option is the default if no other options are entered with the **db2osconf** command. On the Solaris operating system, only the kernel parameters that differ will be displayed. Since the current kernel parameters are taken directly from the live kernel, they might not match those in `/etc/system`, the Solaris system specification file. If the kernel parameters from the live kernel are different than those listed in the `/etc/system`, the `/etc/system` file might have been changed without a reboot or there might be a syntax error in the file. On HP-UX, the `-f` option returns a list of recommended parameters and a list of recommended changes to parameter values:  

```
***** Please Change the Following in the Given Order *****

WARNING [<parameter name>] should be set to <value>
```
- l** The `'-l'` switch lists the current kernel parameters.
- m** The `'-m'` switch overrides the amount of physical memory in GB. Normally, the `db2osconf` utility determines the amount of physical memory automatically. This option is available only on DB2 for the Solaris operating system.
- n** The `'-n'` switch overrides the number of CPUs on the system. Normally, the `db2osconf` utility determines the number of CPUs automatically. This option is available only on DB2 for the Solaris operating system.

## db2osconf - Utility for Kernel Parameter Values

- p The '-p' switch sets the performance level for SYSV message queues. 0 (zero) is the default and 3 is the highest setting. Setting this value higher can increase the performance of the message queue facility at the expense of using more memory.
- s The '-s' switch sets the scale factor. The default scale factor is 1 and should be sufficient for almost any workload. If a scale factor of 1 is not enough, the system might be too small to handle the workload. The scale factor sets the kernel parameters recommendations to that of a system proportionally larger than the size of the current system. For example, a scale factor of 2.5 would recommend kernel parameters for a system that is 2.5 times the size of the current system.
- t The '-t' switch provides recommendations for `semsys:seminfo_semume` and `shmsys:shminfo_shmseg` kernel parameter values. This option is available only on DB2 for the Solaris operating system. For multi-threaded programs with a fair number of connections, these kernel parameters might have to be set beyond their default values. They only need to be reset if the multi-threaded program requiring them is a local application:

### **semsys:seminfo\_semume**

Limit of semaphore undo structures that can be used by any one process

### **shmsys:shminfo\_shmseg**

Limit on the number of shared memory segments that any one process can create.

These parameters are set in the `/etc/system` file. The following is a guide to set the values, and is what the **db2osconf** utility uses to recommend them. For each local connection DB2 will use one semaphore and one shared memory segment to communicate. If the multi-threaded application is a local application and has X number of connections to DB2, then that application (process) will need X number of shared memory segments and X number of the semaphore undo structures to communicate with DB2. So the value of the two kernel Parameters should be set to X + 10 (the plus 10 provides a safety margin).

Without the '-l' or '-f' switches, the **db2osconf** utility displays the kernel parameters using the syntax of the `/etc/system` file. To prevent human errors, the output can be cut and pasted directly into the `/etc/system` file.

The kernel parameters are recommended based on both the number of CPUs and the amount of physical memory on the system. If one is unproportionately low, the recommendations will be based on the lower of the two.

### **Examples:**

Here is a sample output produced by running the **db2osconf** utility with the `-t` switch set for 500 threads. The results received are machine-specific, so the results you receive will vary depending on your environment.

```
db2osconf -t 500

set msgsys:msginfo_msgmax = 65535
set msgsys:msginfo_msgmnb = 65535
set msgsys:msginfo_msgssz = 32
set msgsys:msginfo_msgseg = 32767
set msgsys:msginfo_msgmap = 2562
set msgsys:msginfo_msgmni = 2560
set msgsys:msginfo_msgtql = 2560
```



## db2osconf - Utility for Kernel Parameter Values

```
set semsys:seminfo_semmap = 3074
set semsys:seminfo_semmni = 3072
set semsys:seminfo_semmns = 6452
set semsys:seminfo_semmnu = 3072
set semsys:seminfo_semume = 600
set shmsys:shminfo_shmmax = 2134020096
set shmsys:shminfo_shmmni = 3072
set shmsys:shminfo_shmseg = 600
```

```
Total kernel space for IPC:
0.35MB (shm) + 1.77MB (sem) + 1.34MB (msg) == 3.46MB (total)
```

The recommended values for `set semsys:seminfo_semume` and `set shmsys:shminfo_shmseg` were the additional values provided by running `db2osconf -t 500`.

### Usage notes:

Even though it is possible to recommend kernel parameters based on a particular DB2 workload, this level of accuracy is not beneficial. If the kernel parameter values are too close to what are actually needed and the workload changes in the future, DB2 might encounter a problem due to a lack of interprocess communication (IPC) resources. A lack of IPC resources can lead to an unplanned outage for DB2 and a reboot would be necessary in order to increase kernel parameters. By setting the kernel parameters reasonably high, it should reduce or eliminate the need to change them in the future. The amount of memory consumed by the kernel parameter recommendations is almost trivial compared to the size of the system. For example, for a system with 4GB of RAM and 4 CPUs, the amount of memory for the recommended kernel parameters is 4.67MB or 0.11%. This small fraction of memory used for the kernel parameters should be acceptable given the benefits.

On the Solaris operating system, there are two versions of the **db2osconf** utility: one for 64-bit kernels and one for 32-bit kernels. The utility needs to be run as root or with the group `sys` since it accesses the following special devices (accesses are read-only):

```
crw-r----- 1 root sys 13, 1 Jul 19 18:06 /dev/kmem
crw-rw-rw- 1 root sys 72, 0 Feb 19 1999 /dev/ksyms
crw-r----- 1 root sys 13, 0 Feb 19 1999 /dev/mem
```

### Related tasks:

- “Modifying kernel parameters (Solaris Operating Environment)” in *Quick Beginnings for DB2 Servers*

### Related reference:

- “Recommended kernel configuration parameters (HP-UX)” in *Quick Beginnings for DB2 Servers*

## db2pd - Monitor and troubleshoot DB2 database

The db2pd utility retrieves information from the DB2 database system memory sets.

### Authorization:

One of the following:

- On Linux and UNIX, the *sysadm* authority level. You must also be the instance owner.
- On Windows operating systems, the *sysadm* authority level.

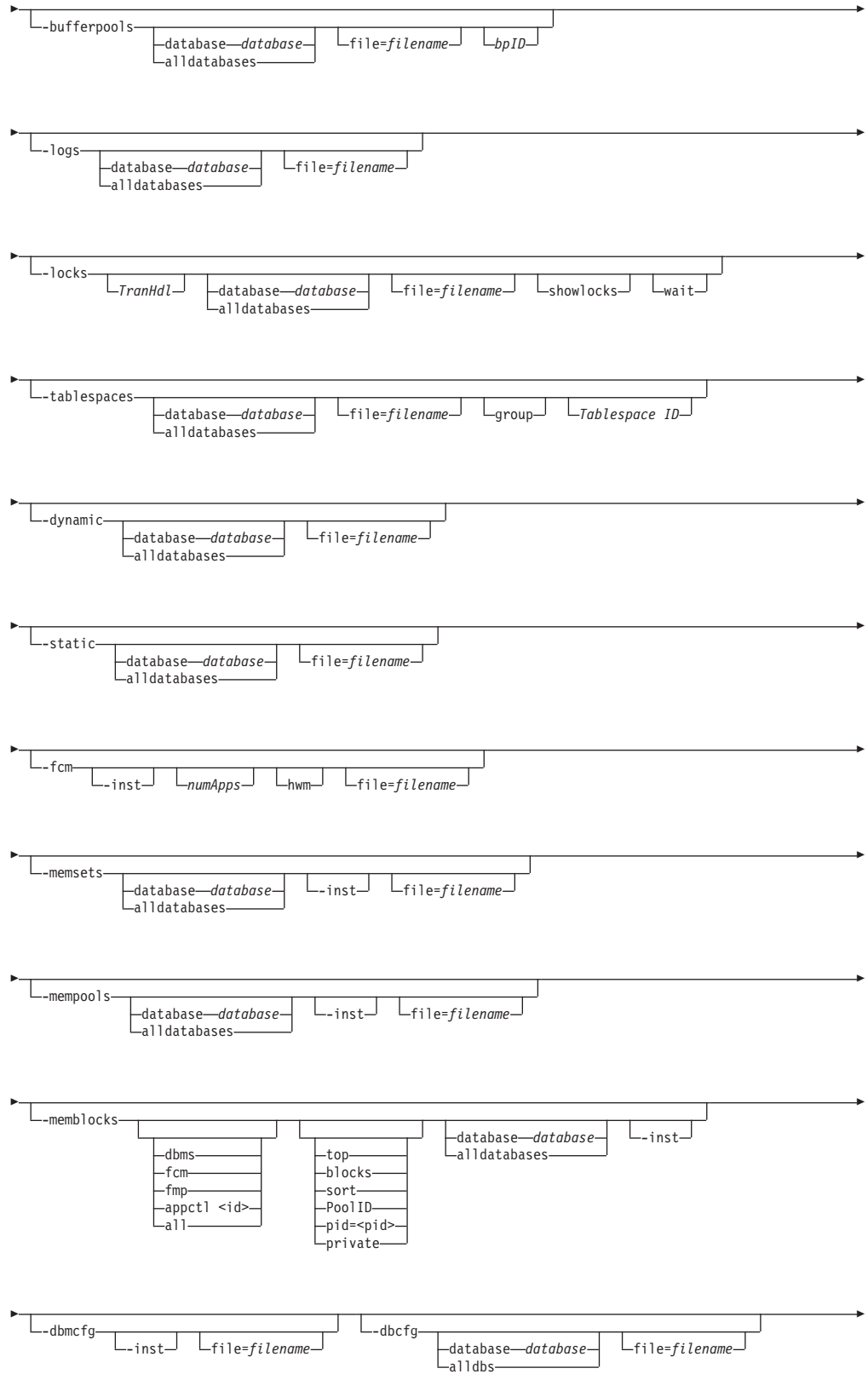
### Required connection:

There is no minimum connection requirement. However, if a database scope option is specified, that database must be active before the command can return the requested information.

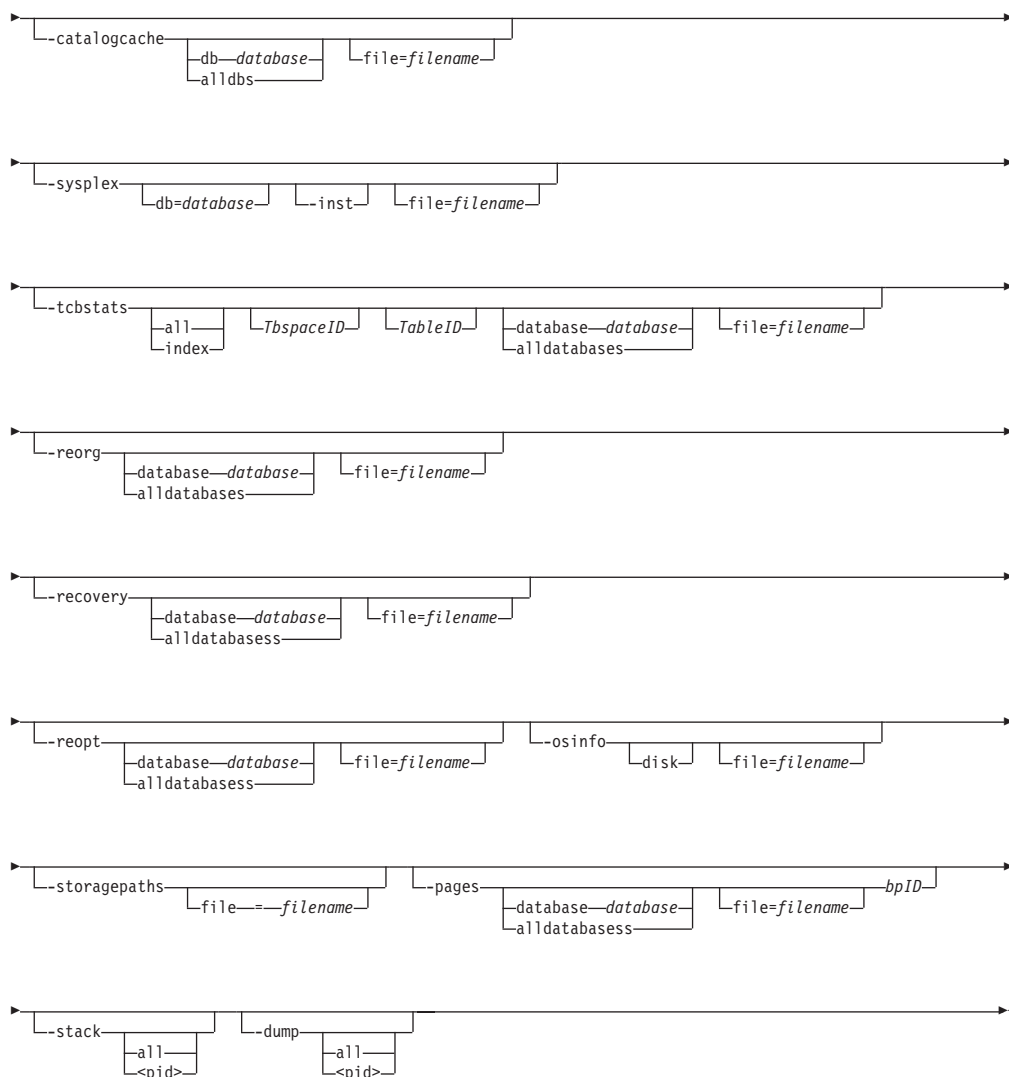
### Command syntax:



## db2pd - Monitor and troubleshoot DB2 database



## db2pd - Monitor and troubleshoot DB2 database



### Command parameters:

- inst** Returns all instance-scope information.
- help** Displays the online help information.
- version** Displays the current version and service level of the installed DB2 product.
- dbpartitionnum** *num* Specifies that the command is to run on the specified database partition server.
- alldbpartitionnums** Specifies that this command is to run on all active database partition servers in the instance. **db2pd** will only report information from database partition servers on the same physical machine that **db2pd** is being run on.
- database** *database* Specifies that the command attaches to the database memory sets of the specified database.

## db2pd - Monitor and troubleshoot DB2 database

### **-alldatabases**

Specifies that the command attaches to all memory sets of all the databases.

### **-everything**

Runs all options for all databases on all database partition servers that are local to the server.

### **-file *filename***

Specifies to write the output to the specified file.

### **-command *filename***

Specifies to read and execute the **db2pd** command options that are specified in the file.

### **-interactive**

Specifies to override the values specified for the DB2PDOPT environment variable when running the **db2pd** command.

**-full** Specifies that all output is expanded to its maximum length. If not specified, output is truncated to save space on the display.

**-hadr** Reports high availability disaster recovery (HADR) information. Descriptions of each reported element can be found in the high availability disaster recovery section of the *System Monitor Guide and Reference*.

### **-utilities**

Reports utility information. Descriptions of each reported element can be found in the utilities section of the *System Monitor Guide and Reference*.

### **-repeat *num sec count***

Specifies that the command is to be repeated after the specified number of seconds. If a value is not specified for the number of seconds, the command repeats every five seconds. You can also specify the number of times the output will be repeated. If you do not specify a value for *count*, the command is repeated until it is interrupted.

### **-applications**

Returns information about applications.

If an application ID is specified, information is returned about that application.

If an agent ID is specified, information is returned about the agent that is working on behalf of the application.

**-fmp** Returns information about the process in which the fenced routines are executed.

### **-agents**

Returns information about agents.

If an agent ID is specified, information is returned about the agent.

If an application ID is specified, information is returned about all the agents that are performing work for the application.

Specify this option with the **-inst** option, if you have chosen a database that you want scope output for.

### **-transactions**

Returns information about active transactions.

If a transaction handle is specified, information is returned about that transaction handle.

## db2pd - Monitor and troubleshoot DB2 database

If an application handle is specified, information is returned about the application handle of the transaction.

### **-bufferpools**

Returns information about the buffer pools. If a bufferpool ID is specified, information is returned about the bufferpool.

**-logs** Returns information about the log files.

**-locks** Returns information about the locks.

Specify a transaction handle to obtain information about the locks that are held by a specific transaction.

Specify this option with the `showlocks` option to return detailed information about lock names. For row and block locks on partitioned tables and individual data partitions, `showlocks` displays the data partition identifier as part of the row with the lock information.

Specify the `wait` option to return locks in a wait state and the owners of those locks.

### **-tablespaces**

Returns information about the table spaces.

Specify this option with the `group` option to display the information about the containers of a table space grouped with the table space.

Specify this option with the `tablespace` option to display the information about a specific table space and its containers.

### **-dynamic**

Returns information about the execution of dynamic SQL.

**-static** Returns information about the execution of static SQL and packages.

**-fcm** Returns information about the fast communication manager.

- Specify this option with the `-inst` option, if you have chosen a database for which you want scope output.
- Specify this option with the `hwm` option, to retrieve high-watermark consumptions of FCM buffers and channels by applications since the start of the DB2 instance. The high-watermark consumption values of applications are retained even if they have disconnected from the database already.
- Specify this option with the `numApps` option, to limit the maximum number of applications that the **db2pd** command reports in the current and HWM consumption statistics.

### **-memsets**

Returns information about the memory sets.

Specify this option with the `-inst` option to include all the instance-scope information in the returned information.

### **-mempools**

Returns information about the memory pools.

Specify this option with the `-inst` option to include all the instance-scope information in the returned information.

### **-memblocks**

Returns information about the memory pools.

## db2pd - Monitor and troubleshoot DB2 database

- Specify this option with the `dbms` option to only report blocks in the dbms memory set.
- Specify this option with the `fcm` option to only report blocks in the fast communication manager memory set.
- Specify this option with the `fmp` option to only report blocks in the fenced mode procedure memory set.
- Specify this option with the `appctl <id>` option to only report blocks in the application control set.
- Specify this option with the `all` option to report blocks from all memory sets.
- Specify this option with the `top` option to report the top memory consumers for each set.
- Specify this option with the `blocks` option to report the memory blocks for each set.
- Specify this option with the `sort` option to report the sorted memory blocks for each pool in each set.
- Specify this option with the `PoolID` option to report memory blocks from a specific pool.
- Specify this option with the `pid=<pid>` option to report memory blocks from a specific process id. (for UNIX operating systems only)
- Specify this option with the `private` option to report memory blocks from the private memory set. (Windows operating systems only)

### **-dbmcfg**

Returns the settings of the database manager configuration parameters.

Specify this option with the `-inst` option, if you have chosen a database for which you want scope output.

**-dbcfg** Returns the settings of the database configuration parameters.

### **-catalogcache**

Returns information about the catalog cache.

### **-sysplex**

Returns information about the list of servers associated with the database alias indicated by the `db` parameter. If the `-database` parameter is not specified, information is returned for all databases.

Specify this option with the `-inst` option, if you have chosen a database for which you want scope output.

### **-tcbstats**

Returns information about tables and indexes. Specify this option with the `TbpaceID` option to display the information about a specific table space.

Specify this option with the `TableID` option to display the information about a specific table. The `TbpaceID` option is required when using the `TableID` option.

**-reorg** Returns information about table and data partition reorganization.

### **-recovery**

Returns information about recovery activity.

**-reopt** Returns information about cached SQL statements that were reoptimized using the `REOPT ONCE` option.

## db2pd - Monitor and troubleshoot DB2 database

### **-osinfo**

Returns operating system information. If a disk path is specified, information about the disk will be printed.

### **-storagepaths**

Returns information about the automatic storage paths defined for the database.

**-pages** Returns information about the buffer pool pages. If bufferpool ID is specified, only pages from the specified bufferpool are returned.

**-stack** Produces stack trace files in the DIAGPATH directory.

- Specify this option with the `all` option to produce stack trace files for all processes in the current database partition.
- Specify this option with the `<pid>` option to produce a stack trace file for a specific process id.

### **-dump**

Produces stack trace and binary dump files in the DIAGPATH directory.

- Specify this option with the `all` option to produce stack trace files and binary dump files for all agents in the current database partition.
- Specify this option with the `<pid>` option to produce a stack trace file and binary dump file for a specific agent.

### **Examples:**

Use the **db2pd** command from the command line in the following way to obtain information about agents that are servicing client requests:

```
db2pd -agents
```

Use the **db2pd** command from the command line in the following way to obtain information about agents that are servicing client requests. In this case, the `DB2PDOPT` environment variable is set with the `-agents` parameter before invoking the **db2pd** command. The command uses the information set in the environment variable when it executes.

```
export DB2PDOPT="-agents"
db2pd
```

Use the **db2pd** command from the command line in the following way to obtain information about agents that are servicing client requests. In this case, the `-agents` parameter is set in the file `file.out` before invoking the **db2pd** command. The **-command** parameter causes the command to use the information in the `file.out` file when it executes.

```
echo "-agents" > file.out
db2pd -command file.out
```

Use the **db2pd** command from the command line in the following way to obtain all database and instance-scope information:

```
db2pd -inst -alldbs
```

### **Usage notes:**

The following sections describe the output produced by the different **db2pd** parameters.

- “-applications” on page 191
- “-fmp” on page 191
- “-agents” on page 192



- “-transactions” on page 192
- “-bufferpools” on page 194
- “-logs” on page 197
- “-locks” on page 198
- “-tablespaces” on page 199
- “-dynamic” on page 203
- “-static” on page 204
- “-fcm” on page 205
- “-memsets” on page 207
- “-mempools” on page 207
- “-memblocks” on page 208
- “-dbmcfg” on page 209
- “-dbcfg” on page 209
- “-catalogcache” on page 209
- “-sysplex” on page 212
- “-tcbstats” on page 212
- “-reorg” on page 215
- “-recovery” on page 217
- “-reopt” on page 218
- “-osinfo” on page 218
- “-storagepaths” on page 221
- “-pages” on page 221

### **-applications parameter:**

For the -applications parameter, the following information is returned:

#### **ApplHandl**

The application handle, including the node and the index.

#### **NumAgents**

The number of agents that are working on behalf of the application.

#### **CoorPid**

The process ID of the coordinator agent for the application.

**Status** The status of the application.

**Appid** The application ID.

### **-fmp parameter:**

For the -fmp parameter, the following information is returned:

- Pool Size - Current number of FMP processes in the FMP pool.
- Max Pool Size - Maximum number of FMP process in the FMP pool.
- Keep FMP - Value of KEEPFENCED database manager configuration parameter.
- Initialized - FMP is initialized. Possible values are Yes and No.
- Trusted Path - Path of trusted procedures
- Fenced User - Fenced user ID

#### **FMP Process:**

- FmpPid - Process ID of the FMP process.
- Bit - Bit mode. Values are 32 bit or 64 bit.
- Flags - State flags for the FMP process. Possible values are:
  - 0x00000000 - JVM initialized
  - 0x00000002 - Is threaded

## db2pd - Monitor and troubleshoot DB2 database

- 0x00000004 - Used to run federated wrappers
- 0x00000008 - Used for Health Monitor
- 0x00000010 - Marked for shutdown and will not accept new tasks
- 0x00000020 - Marked for cleanup by db2sysc
- 0x00000040 - Marked for agent cleanup
- 0x00000100 - All ipc's for the process have been removed
- 0x00000200 - .NET runtime initialized
- 0x00000400 - JVM initialized for debugging
- 0x00000800 - Termination flag
- ActiveTh - Number of active threads running in the fmp process.
- PooledTh - Number of pooled threads held by the fmp process.
- Active - Active state of the fmp process. Values are Yes or No.

### Active Threads:

- FmpPid - FMP process ID that owns the active thread.
- EduPid - EDU process ID that this thread is working.
- ThreadId - Active thread ID.

### Pooled Threads:

- FmpPid - FMP process ID that owns the pooled thread.
- ThreadId - Pooled thread ID.

### -agents parameter:

For the -agents parameter, the following information is returned:

#### AppHandl

The application handle, including the node and the index.

#### AgentPid

The process ID of the agent process.

#### Priority

The priority of the agent.

**Type** The type of agent.

**State** The state of the agent.

#### ClientPid

The process ID of the client process.

#### Userid

The user ID running the agent.

#### ClientNm

The name of the client process.

#### Rowsread

The number of rows that were read by the agent.

#### Rowswrtn

The number of rows that were written by the agent.

#### LkTmOt

The lock timeout setting for the agent.

### -transactions parameter:

For the `-transactions` parameter, the following information is returned:

**ApplHandl**

The application handle of the transaction.

**TranHdl**

The transaction handle of the transaction.

**Locks** The number of locks held by the transaction.

**State** The transaction state.

**Tflag** The transaction flag. The possible values are:

- 0x00000002. This value is only written to the coordinator node of a two-phase commit application, and it indicates that all subordinate nodes have sent a "prepare to commit" request.
- 0x00000020. The transaction must change a capture source table (used for data replication only).
- 0x00000040. Crash recovery considers the transaction to be in the prepare state.
- 0x00010000. This value is only written to the coordinator partition in a partitioned database environment, and it indicates that the coordinator partition has not received a commit request from all subordinate partitions in a two-phase commit transaction.
- 0x00040000. The rolling back of the transaction is pending.
- 0x01000000. The transaction resulted in an update on a database partition server that is not the coordinator partition.
- 0x04000000. Loosely coupled XA transactions are supported.
- 0x08000000. Multiple branches are associated with this transaction and are using the loosely coupled XA protocol.
- 0x10000000. A data definition language (DDL) statement has been issued, indicating that the loosely coupled XA protocol cannot be used by the branches participating in the transaction.

**Tflag2** Transaction flag 2. The possible values are:

- 0x00000004. The transaction has exceeded the limit specified by the `num_log_span` database configuration parameter.
- 0x00000008. The transaction resulted because of the running of a DB2 utility.
- 0x00000020. The transaction will cede its locks to an application with a higher priority (this value ordinarily occurs for jobs that the DB2 database system automatically starts for self tuning and self management).
- 0x00000040. The transaction will not cede its row-level locks to an application with a higher priority (this value ordinarily occurs for jobs that the DB2 database system automatically starts for self-tuning and self-management)

**Firstlsn**

First LSN of the transaction.

**Lastlsn**

Last LSN of the transaction.

**LogSpace**

The amount of log space that is reserved for the transaction.

## db2pd - Monitor and troubleshoot DB2 database

### SpaceReserved

The total log space that is reserved for the transaction, including the used space and all compensation records.

**TID** Transaction ID.

### AxRegCnt

The number of applications that are registered for a global transaction. For local transactions, the value is 1.

**GXID** Global transaction ID. For local transactions, the value is 0.

### -bufferpools parameter:

For the -bufferpools parameter, the following information is returned:

#### First Active Pool ID

The ID of the first active buffer pool.

#### Max Bufferpool ID

The maximum ID of all active buffer pools.

#### Max Bufferpool ID on Disk

The maximum ID of all buffer pools defined on disk.

#### Num Bufferpools

The number of available buffer pools.

**ID** The ID of the buffer pool.

**Name** The name of the buffer pool.

#### PageSz

The size of the buffer pool pages.

#### PA-NumPgs

The number of pages in the page area of the buffer pool.

#### BA-NumPgs

The number of pages in the block area of the buffer pool. This value is 0 if the buffer pool is not enabled for block-based I/O.

#### BlkSize

The block size of a block in the block area of the buffer pool. This value is 0 if the buffer pool is not enabled for block-based I/O.

#### NumTbsp

The number of table spaces that are using the buffer pool.

#### PgsLeft

The number of pages left to remove in the buffer pool if its size is being decreased.

#### CurrentSz

The current size of the buffer pool in pages.

#### PostAlter

The size of the buffer pool in pages when the buffer pool is restarted.

#### SuspndTSCt

The number of table spaces mapped to the buffer pool that are currently I/O suspended. If 0 is returned for all buffer pools, the database I/O is not suspended.

### **DatLRds**

Buffer Pool Data Logical Reads. Indicates the number of data pages which have been requested from the buffer pool (logical) for regular and large table spaces.

### **DatPRds**

Buffer Pool Data Physical Reads. Indicates the number of data pages read in from the table space containers (physical) for regular and large table spaces.

### **HitRatio**

Hit ratio for data pages in the buffer pool using formula  $1 - \text{DatPRds} / \text{DatLRds}$ .

### **TmpDatLRds**

Buffer Pool Temporary Data Logical Reads. Indicates the number of data pages which have been requested from the buffer pool (logical) for temporary table spaces.

### **TmpDatPRds**

Buffer Pool Temporary Data Physical Reads. Indicates the number of data pages read in from the table space containers (physical) for temporary table spaces.

### **HitRatio**

Hit ratio for temporary data pages in the buffer pool using formula  $1 - \text{TmpDatPRds} / \text{TmpDatLRds}$ .

### **IdxLRds**

Buffer Pool Index Logical Reads. Indicates the number of index pages which have been requested from the buffer pool (logical) for regular and large table spaces.

### **IdxPRds**

Buffer Pool Index Physical Reads. Indicates the number of index pages read in from the table space containers (physical) for regular and large table spaces.

### **HitRatio**

Hit ratio for index pages in the buffer pool using formula  $1 - \text{IdxPRds} / \text{IdxLRds}$ .

### **TmpIdxLRds**

Buffer Pool Temporary Index Logical Reads. Indicates the number of index pages which have been requested from the buffer pool (logical) for temporary table spaces.

### **TmpIdxPRds**

Buffer Pool Temporary Index Physical Reads. Indicates the number of index pages read in from the table space containers (physical) for temporary table spaces.

### **HitRatio**

Hit ratio for temporary index pages in the buffer pool using formula  $1 - \text{TmpIdxPRds} / \text{TmpIdxLRds}$ .

### **DataWrts**

Buffer Pool Data Writes. Indicates the number of times a buffer pool data page was physically written to disk.

## db2pd - Monitor and troubleshoot DB2 database

### **IdxWrts**

Buffer Pool Index Writes. Indicates the number of times a buffer pool index page was physically written to disk.

### **DirRds**

Direct Reads From Database. The number of read operations that do not use the buffer pool.

### **DirRdReqs**

Direct Read Requests. The number of requests to perform a direct read of one or more sectors of data.

### **DirRdTime**

Direct Read Time. The elapsed time (in milliseconds) required to perform the direct reads.

### **DirWrts**

Direct Writes to Database. The number of write operations that do not use the buffer pool.

### **DirWrtReqs**

Direct Write Requests. The number of requests to perform a direct write of one or more sectors of data.

### **DirWrtTime**

Direct Write Time. The elapsed time (in milliseconds) required to perform the direct writes.

### **AsDatRds**

Buffer Pool Asynchronous Data Reads. Indicates the number of data pages read in from the table space containers (physical) by asynchronous engine dispatchable units (EDUs) for all types of table spaces.

### **AsDatRdReq**

Buffer Pool Asynchronous Read Requests. The number of asynchronous read requests.

### **AsIdxRds**

Buffer Pool Asynchronous Index Reads. Indicates the number of index pages read in from the table space containers (physical) by asynchronous engine dispatchable units (EDUs) for all types of table spaces.

### **AsIdxRdReq**

Buffer Pool Asynchronous Index Read Requests. The number of asynchronous read requests for index pages.

### **AsRdTime**

Buffer Pool Asynchronous Read Time. Indicates the total amount of time spent reading in data and index pages from the table space containers (physical) by asynchronous engine dispatchable units (EDUs) for all types of table spaces. This value is given in microseconds.

### **AsDatWrts**

Buffer Pool Asynchronous Data Writes. The number of times a buffer pool data page was physically written to disk by either an asynchronous page cleaner, or a prefetcher. A prefetcher might have written dirty pages to disk to make space for the pages being prefetched.

### **AsIdxWrts**

Buffer Pool Asynchronous Index Writes. The number of times a buffer pool index page was physically written to disk by either an asynchronous page

## db2pd - Monitor and troubleshoot DB2 database

cleaner, or a prefetcher. A prefetcher might have written dirty pages to disk to make space for the pages being prefetched.

### **AsWrtTime**

Buffer Pool Asynchronous Write Time. The total elapsed time spent writing data or index pages from the buffer pool to disk by database manager page cleaners.

### **TotRdTime**

Total Buffer Pool Physical Read Time. Indicates the total amount of time spent reading in data and index pages from the table space containers (physical) for all types of table spaces. This value is given in microseconds.

### **TotWrtTime**

Total Buffer Pool Physical Write Time. Provides the total amount of time spent physically writing data or index pages from the buffer pool to disk. Elapsed time is given in microseconds.

### **VectIORds**

Total Number of Pages Read by Vectored IO. The total number of pages read by vectored I/O into the page area of the buffer pool.

### **VectIOReq**

Number of Vectored IO Requests. The number of vectored I/O requests. More specifically, the number of times the DB2 database product performs sequential prefetching of pages into the page area of the buffer pool.

### **BlockIORds**

Total Number of Pages Read by Block IO. The total number of pages read by block I/O into the block area of the bufferpool.

### **BlockIOReq**

Number of Block IO Requests. The number of block I/O requests. More specifically, the number of times the DB2 database product performs sequential prefetching of pages into the block area of the bufferpool.

### **PhyPgMaps**

Number of Physical Page Maps. The number of physical page maps.

### **FilesClose**

Database Files Closed. The total number of database files closed.

### **NoVictAvl**

Buffer Pool No Victim Buffers. Number of times an agent did not have a preselected victim buffer available.

### **UnRdPFetch**

Unread Prefetch Pages. Indicates the number of pages that the prefetcher read in that were never used.

### **-logs parameter:**

For the -logs parameter, the following information is returned:

#### **Current Log Number**

The number of the current active log.

#### **Method 1 Archive Status**

The result of the most recent log archive attempt. Possible values are Success or Failure.

#### **Method 1 Next Log to Archive**

The next log file to be archived.

## db2pd - Monitor and troubleshoot DB2 database

### Method 1 First Failed

The first log file that was unsuccessfully archived.

### Method 2 Archive Status

The result of the most recent log archive attempt. Possible values are Success or Failure.

### Method 2 Next Log to Archive

The next log file to be archived.

### Method 2 First Failed

The first log file that was unsuccessfully archived.

### Pages Written

The current page being written in the current log.

### StartLSN

The starting log sequence number.

**State** 0x00000020 indicates that the log has been archived.

**Size** The size of the log's extent, in pages.

**Pages** The number of pages in the log.

### Filename

The file name of the log.

### -locks parameter:

For the -locks parameter, the following information is returned:

### TranHdl

The transaction handle that is requesting the lock.

### Lockname

The name of the lock.

**Type** The type of lock. The possible values are:

- Row
- Pool
- Partition
- Table
- AlterTab
- ObjectTab
- OnlBackup
- DMS Seq
- Internal P
- Internal V
- Key Value
- No Lock
- Block Lock
- LOG Release
- LF Release
- LFM File
- LOB/LF 4K
- APM Seq
- Tbsp Load
- Table Part
- DJ UserMap
- DF NickNm
- CatCache



- OnlReorg
- Buf Pool

**Mode** The lock mode. The possible values are:

- no lock
- IS
- IX
- S
- SIX
- X
- IN
- Z
- U
- NS
- NX
- W
- NW

**Sts** The lock status. The possible values are:

- G (granted)
- C (converting)
- W (waiting)

**Owner**

The transaction handle that owns the lock.

**Dur** The duration of the lock.

**HoldCount**

The number of holds placed on the lock. Locks with holds are not released when transactions are committed.

**Att** The attributes of the lock. Possible values are:

- 0x01 Wait for availability.
- 0x02 Acquired by escalation.
- 0x04 RR lock "in" block.
- 0x08 Insert Lock.
- 0x10 Lock by RR scan.
- 0x20 Update/delete row lock.
- 0x40 Allow new lock requests.
- 0x80 A new lock requestor.

**ReleaseFlg**

The lock release flags. Possible values are:

- 0x80000000 Locks by SQL compiler.
- 0x40000000 Non-unique, untracked locks.

**-tablespaces parameter:**

For the -tablespaces parameter, the output is organized into four segments:

**Table space Configuration:**

**Id** The table space ID.

**Type** The type of table space. The possible values are:

- SMS
- DMS

## db2pd - Monitor and troubleshoot DB2 database

### **Content**

The type of content. The possible values are:

- Regular
- Large
- SysTmp
- UsrTmp

### **PageSz**

The page size used for the table space.

### **ExtentSz**

The size of an extent in pages.

**Auto** Indicates whether the prefetch size is set to AUTOMATIC. The possible values are:

- Yes
- No

### **Prefetch**

The number of pages read from the table space for each range prefetch request.

**BufID** The ID of the buffer pool that this table space is mapped to.

### **BufIDDisk**

The ID of the buffer pool that this table space will be mapped to at next startup.

**FSC** File system caching, indicates whether buffered I/O was specified by the user at CREATE/ALTER TABLESPACE time. The possible values are:

- Yes
- No

### **NumCntns**

The number of containers owned by a table space.

### **MaxStripe**

The maximum stripe set currently defined in the table space (applicable to DMS table spaces only).

### **LastConsecPg**

The last consecutive object table extent.

**Name** The name of the table space.

### **Table space Statistics:**

**Id** The table space ID.

### **TotalPages**

For DMS table spaces, the sum of the gross size of each of the table space's containers (reported in the total pages field of the container).

For SMS table spaces, this value reflects the number of pages in the filesystem owned by the table space.

### **UsablePgs**

For DMS table spaces, the sum of the net size of each of the table space's containers (reported in the usable pages field of the container).

For SMS table spaces, this value reflects the number of pages in the filesystem owned by the table space.

### UsedPgs

For DMS table spaces, the total number of pages currently in use in the table space.

For SMS table spaces, this value reflects the number of pages in the filesystem owned by the table space.

### PndFreePgs

The number of pages that are not available for use but will be available if all the currently outstanding transactions commit.

### FreePgs

For DMS table spaces, the number of pages available for use in the table space.

For SMS table spaces, this value is always 0.

**HWM** The highest allocated page in the table space.

### State

- 0x0000000 - NORMAL
- 0x0000001 - QUIESCED: SHARE
- 0x0000002 - QUIESCED: UPDATE
- 0x0000004 - QUIESCED: EXCLUSIVE
- 0x0000008 - LOAD PENDING
- 0x0000010 - DELETE PENDING
- 0x0000020 - BACKUP PENDING
- 0x0000040 - ROLLFORWARD IN PROGRESS
- 0x0000080 - ROLLFORWARD PENDING
- 0x0000100 - RESTORE PENDING
- 0x0000200 - DISABLE PENDING
- 0x0000400 - REORG IN PROGRESS
- 0x0000800 - BACKUP IN PROGRESS
- 0x0001000 - STORAGE MUST BE DEFINED
- 0x0002000 - RESTORE IN PROGRESS
- 0x0004000 - OFFLINE
- 0x0008000 - DROP PENDING
- 0x0010000 - WRITE SUSPENDED
- 0x0020000 - LOAD IN PROGRESS
- 0x0200000 - STORAGE MAY BE DEFINED
- 0x0400000 - STORAGE DEFINITION IS IN FINAL STATE
- 0x0800000 - STORAGE DEFINITION CHANGED PRIOR TO ROLLFORWARD
- 0x1000000 - DMS REBALANCER IS ACTIVE
- 0x2000000 - DELETION IN PROGRESS
- 0x4000000 - CREATION IN PROGRESS

### MinRecTime

The minimum recovery time for the table space.

### NQuiescers

The number of quiescers.

### Table space Autoresize Statistics:

**Id** The table space ID.

**AS** Indicates whether or not the table space is using automatic storage. The possible values are:

- Yes
- No

## db2pd - Monitor and troubleshoot DB2 database

**AR** Indicates whether or not the table space is enabled to be automatically resized. The possible values are:

- Yes
- No

**InitSize**

For automatic storage table spaces, the value of this parameter is the initial size of the table space in bytes.

**IncSize**

For automatically resized table spaces, if the value of the IIP field is No, the value of this parameter is the size, in bytes, that the table space will automatically be increased by (per database partition) when the table space is full and a request for space is made. If the value of the IIP field is Yes, the value of this parameter is a percentage.

**IIP** For automatically resized table spaces, the value of this parameter indicates whether the increment value in the IncSize field is a percent or not. The possible values are:

- Yes
- No

**MaxSize**

For automatically resized table spaces, the value of this parameter specifies the maximum size, in bytes, to which the table space can automatically be increased (per database partition). A value of NONE indicates that there is no maximum size.

**LastResize**

The timestamp of the last successful automatic resize operation.

**LRF** Last resize failed indicates whether the last automatic resizing operation was successful or not. The possible values are:

- Yes
- No

**Table space Containers:**

**TspId** The ID of the table space that owns the container.

**ContainNum**

The number assigned to the container in the table space.

**Type** The type of container. The possible values are:

- Path
- Disk
- File
- Striped Disk
- Striped File

**TotalPgs**

The number of pages in the container.

**UsablePgs**

The number of usable pages in the container.

**StripeSet**

The stripe set where the container resides (applicable to DMS table spaces only).

**Container**

The name of the container.

### -dynamic parameter:

For the -dynamic parameter, the following information is returned:

#### Dynamic Cache:

**Current Memory Used**

The number of bytes used by the package cache.

**Total Heap Size**

The number of bytes configured internally for the package cache.

**Cache Overflow flag state**

A flag to indicate whether the package cache is in an overflow state.

**Number of references**

The number of times the dynamic portion of the package cache has been referenced.

**Number of Statement Inserts**

The number of statement inserts into the package cache.

**Number of Statement Deletes**

The number of statement deletions from the package cache.

**Number of Variation Inserts**

The number of variation inserts into the package cache.

**Number of statements**

The number of statements in the package cache.

#### Dynamic SQL Statements:

**AnchID**

The hash anchor identifier.

**StmtID**

The statement identifier.

**NumEnv**

The number of environments that belong to the statement.

**NumVar**

The number of variations that belong to the statement.

**NumRef**

The number of times that the statement has been referenced.

**NumExe**

The number of times that the statement has been executed.

**Text**

The text of the SQL statement.

#### Dynamic SQL Environments:

**AnchID**

The hash anchor identifier.

**StmtID**

The statement identifier.

**EnvID** The environment identifier.

**Iso** The isolation level of the environment.

**QOpt** The query optimization level of the environment.

## db2pd - Monitor and troubleshoot DB2 database

**Blk** The blocking factor of the environment.

### Dynamic SQL Variations:

**AnchID**  
The hash anchor identifier.

**StmtID**  
The statement identifier for this variation.

**EnvID** The environment identifier for this variation.

**VarID** The variation identifier.

**NumRef**  
The number of times this variation has been referenced.

**Typ** The internal statement type value for the variation section.

**Lockname**  
The variation lockname.

### -static parameter:

For the -static parameter, the following information is returned:

#### Static Cache:

**Current Memory Used**  
The number of bytes used by the package cache.

**Total Heap Size**  
The number of bytes internally configured for the package cache.

**Cache Overflow flag state**  
A flag to indicate whether the package cache is in an overflow state.

**Number of References**  
The number of references to packages in the package cache.

**Number of Package Inserts**  
The number of package inserts into the package cache.

**Number of Section Inserts**  
The number of static section inserts into the package cache.

#### Packages:

**Schema**  
The qualifier of the package.

**PkgName**  
The name of the package.

**Version**  
The version identifier of the package.

**UniqueID**  
The consistency token associated with the package.

**NumSec**  
The number of sections that have been loaded.

**UseCount**  
The usage count of the cached package.

**NumRef**

The number of times the cached package has been referenced.

**Iso** The isolation level of the package.

**QOpt** The query optimization of the package.

**Blk** The blocking factor of the package.

**Lockname**

The lockname of the package.

**Sections:****Schema**

The qualifier of the package that the section belongs to.

**PkgName**

The package name that the section belongs to.

**UniqueID**

The consistency token associated with the package that the section belongs to.

**SecNo** The section number.

**NumRef**

The number of times the cached section has been referenced.

**UseCount**

The usage count of the cached section.

**StmtType**

The internal statement type value for the cached section.

**Cursor**

The cursor name (if applicable).

**W-Hld**

Indicates whether the cursor is a WITH HOLD cursor.

**-fcm parameter:**

For the -fcm parameter, the following information is returned:

**FCM Usage Statistics:****Total Buffers**

Total number of buffers, including all free and in-use ones.

**Free Buffers**

Number of free buffers.

**Buffers LWM**

Lowest number of free buffers.

**Total Channels**

Total number of channels, including all free and in-use ones.

**Free Channels**

Number of free channels.

**Channels LWM**

Lowest number of free channels.

**Total Sessions**

Total number of sessions, including all free and in-use ones.

## db2pd - Monitor and troubleshoot DB2 database

### Free Sessions

Number of free sessions.

### Sessions LWM

Lowest number of free sessions.

### Partition

The database partition server number.

### Bufs Sent

The total number of FCM buffers that are sent from the database partition server where the **db2pd** command is running to the database partition server that is identified in the output.

### Bufs Recv

The total number of FCM buffers that are received by the database partition server where the **db2pd** command is running from the database partition server that is identified in the output.

**Status** The logical connection status between the database partition server where the **db2pd** command is running and the other database partition servers that are listed in the output. The possible values are:

- Inactive: the database partition server is defined in the Data Partitioning Feature configuration but is currently inactive (for example, the user has stopped the partition).
- Active: the database partition server is active.
- Undefined: the database partition server is not defined in the Data Partitioning Feature configuration. This might indicate an error.
- Unknown: the database partition server is in an unknown state. This indicates an error.

### Buffers Current Consumption

#### AppHandl

The application handle, including the node and the index.

#### TimeStamp

A unique identifier for the usage of an application handle.

#### Buffers In-use

The number of buffers currently being used by an application.

### Channels Current Consumption

#### AppHandl

The application handle, including the node and the index.

#### TimeStamp

A unique identifier for the usage of an application handle.

#### Channels In-use

The number of channels currently being used by an application.

### Buffers Consumption HWM

#### AppHandl

The application handle, including the node and the index.

#### TimeStamp

A unique identifier for the usage of an application handle.



### Buffers Used

The high-watermark number of buffers used by an application since the start of the instance.

### Channels Consumption HWM

#### AppHandl

The application handle, including the node and the index.

#### TimeStamp

A unique identifier for the usage of an application handle.

### Buffers Used

The high-watermark number of channels used by an application since the start of the instance.

### -memsets parameter:

For the -memsets parameter, the following information is returned:

**Name** The name of the memory set.

#### Address

The address of the memory set.

**Id** The memory set identifier.

#### Size(Kb)

The size of the memory set in kilobytes.

**Key** The memory set key (for UNIX based systems only).

**DBP** The database partition server that owns the memory set.

**Type** The type of memory set.

#### Unrsv(Kb)

Memory not reserved for any particular pool. Any pool in the set can use this memory if needed.

#### Used(Kb)

Memory currently allocated to memory pools.

#### Cmt(Kb)

All memory that has been committed by the DB2 database, and occupies physical RAM, paging space, or both.

#### Uncmt(Kb)

Memory not currently being used, and marked by the DB2 database to be uncommitted. Depending on the operating system, this memory could occupy physical RAM, paging space, or both.

### -mempools parameter:

For the -mempools parameter, the following information is returned (All sizes are specified in bytes.):

#### MemSet

The memory set that owns the memory pool.

#### PoolName

The name of the memory pool.

**Id** The memory pool identifier.

## db2pd - Monitor and troubleshoot DB2 database

### **Overhead**

The internal overhead required for the pool structures.

**LogSz** The current total of pool memory requests.

### **LogUpBnd**

The current logical size upper bound.

### **LogHWM**

The logical size high water mark.

**PhySz** The physical memory required for logical size.

### **PhyUpBnd**

The current physical size upper bound.

### **PhyHWM**

The largest physical size reached during processing.

**Bnd** The internal bounding strategy.

### **BlkCnt**

The current number of allocated blocks in the memory pool.

### **CfgParm**

The configuration parameter that declares the size of the pool being reported.

### **-memblocks parameter:**

For the -memblocks parameter, there are three sections of output: individual blocks for the memory set, sorted totals grouped by memory pool, and sorted totals for the memory set:

Memory blocks:

### **PoolID**

The memory pool id that owns the memory block.

### **PoolName**

The memory pool name that owns the memory block.

### **BlockAge**

The block age of the memory block. This is an incremental counter assigned as blocks are allocated.

**Size** The size of the memory block in bytes.

**I** The type of allocation. Value 1 means block will be freed individually while value 0 means it will be freed with the pool.

**LOC** Line of code that allocated the memory block.

**File** Filename hash value from where the block was allocated.

Sorted totals reported for each memory pool:

### **PoolID**

The memory pool id that owns the memory block.

### **PoolName**

The memory pool name that owns the memory block.

**TotalSize**

The total size of blocks (in bytes) allocated from the same line of code and file.

**TotalCount**

The number of blocks allocated from the same line of code and file.

**LOC** Line of code that allocated the memory block.

**File** Filename hash value from where the block was allocated.

Sorted totals reported for each memory set:

**PoolID**

The memory pool id that owns the memory block.

**PoolName**

The memory pool name that owns the memory block.

**TotalSize**

The total size of blocks (in bytes) allocated from the same line of code and file.

**%Bytes**

The percentage bytes allocated from the same line of code and file.

**TotalCount**

The number of blocks allocated from the same line of code and file.

**%Count**

The percentage count allocated from the same line of code and file.

**LOC** Line of code that allocated the memory block.

**File** Filename hash value from where the block was allocated.

**-dbmcfg parameter:**

For the -dbmcfg parameter, current values of the database manager configuration parameters are returned.

**-dbcfg parameter:**

For the -dbcfg parameter, the current values of the database configuration parameters are returned.

**-catalogcache parameter:**

For the -catalogcache parameter, the following information is returned:

**Catalog Cache:****Configured Size**

The number of bytes as specified by the *catalogcache\_sz* database configuration parameter.

**Current Size**

The current number of bytes used in the catalog cache.

**Maximum Size**

The maximum amount of memory that is available to the cache (up to the maximum database global memory).

## db2pd - Monitor and troubleshoot DB2 database

### High Water Mark

The largest physical size reached during processing.

### SYSTABLES:

#### Schema

The schema qualifier for the table.

**Name** The name of the table.

**Type** The type of the table.

#### TableID

The table identifier.

#### TbpaceID

The identifier of the table space where the table resides.

#### LastRefID

The last process identifier that referenced the table.

#### CatalogCache LoadingLock

The name of the catalog cache loading lock for the cache entry.

#### CatalogCache UsageLock

The name of the usage lock for the cache entry.

**Sts** The status of the entry. The possible values are:

- V (valid).
- I (invalid).

### SYSRTNS:

#### RoutineID

The routine identifier.

#### Schema

The schema qualifier of the routine.

**Name** The name of the routine.

#### LastRefID

The last process identifier that referenced the routine.

#### CatalogCache LoadingLock

The name of the catalog cache loading lock for the cache entry.

#### CatalogCache UsageLock

The name of the usage lock for the cache entry.

**Sts** The status of the entry. The possible values are:

- V (valid).
- I (invalid).

### SYSRTNS\_PROCSCHEMAS:

#### RtnName

The name of the routine.

#### ParmCount

The number of parameters in the routine.

#### LastRefID

The last process identifier that referenced the PROCSCHEMAS entry.

### **CatalogCache LoadingLock**

The name of the catalog cache loading lock for the cache entry.

### **CatalogCache UsageLock**

The name of the usage lock for the cache entry.

**Sts** The status of the entry. The possible values are:

- V (valid).
- I (invalid).

### **SYSDATATYPES:**

**TypID** The type identifier.

### **LastRefID**

The last process identifier that referenced the type.

### **CatalogCache LoadingLock**

The name of the catalog cache loading lock for the cache entry.

### **CatalogCache UsageLock**

The name of the usage lock for the cache entry.

**Sts** The status of the entry. The possible values are:

- V (valid).
- I (invalid).

### **SYSCODEPROPERTIES:**

### **LastRefID**

The last process identifier to reference the SYSCODEPROPERTIES entry.

### **CatalogCache LoadingLock**

The name of the catalog cache loading lock for the cache entry.

### **CatalogCache UsageLock**

The name of the usage lock for the cache entry.

**Sts** The status of the entry. The possible values are:

- V (valid).
- I (invalid).

### **SYSNODEGROUPS:**

### **PMapID**

The distribution map identifier.

### **RBalID**

The identifier if the distribution map that was used for the data redistribution.

### **CatalogCache LoadingLock**

The name of the catalog cache loading lock for the cache entry.

### **CatalogCache UsageLock**

The name of the usage lock for the cache entry.

**Sts** The status of the entry. The possible values are:

- V (valid).
- I (invalid).

### **SYSDBAUTH:**

### **AuthID**

The authorization identifier (*authid*).

## db2pd - Monitor and troubleshoot DB2 database

**AuthType**  
The authorization type.

**LastRefID**  
The last process identifier to reference the cache entry.

**CatalogCache LoadingLock**  
The name of the catalog cache loading lock for the cache entry.

### SYSRTNAUTH:

**AuthID**  
The authorization identifier (*authid*).

**AuthType**  
The authorization type.

**Schema**  
The schema qualifier of the routine.

**RoutineName**  
The name of the routine.

**RtnType**  
The type of the routine.

**CatalogCache LoadingLock**  
The name of the catalog cache loading lock for the cache entry.

### -sysplex parameter:

For the -sysplex parameter, the following information is returned:

**Alias** The database alias.

**Location Name**  
The unique name of the database server.

**Count** The number of entries found in the list of servers.

**IP Address**  
The IP address of the server

**Port** The IP port being used by the server.

**Priority**  
The normalized Workload Manager (WLM) weight.

**Connections**  
The number of active connections to this server.

**Status** The status of the connection. The possible values are:

- 0. Healthy.
- 1. Unhealthy. The server is in the list but a connection cannot be established. This entry currently is not considered when establishing connections.
- 2. Unhealthy. The server was previously unavailable, but currently it will be considered when establishing connections.

**PRDID**  
The product identifier of the server as of the last connection.

### -tcbstats parameter:

For the `-tcbstats` parameter, the following information is returned:

### TCB Table Information:

**TbSpaceID**

The table space identifier.

**TableID**

The table identifier.

**PartID**

For partitioned tables, this is the data partition identifier. For non-partitioned table this will display 'n/a'.

**MasterTbs**

For partitioned tables, this is the logical table space identifier to which the partitioned table belongs. For non-partitioned tables, this value corresponds to the `TbSpaceID`.

**MasterTab**

For partitioned tables, this is the logical table identifier of the partitioned table. For non-partitioned tables, this value corresponds to the `TableID`.

**TableName**

The name of the table.

**SchemaNm**

The schema that qualifies the table name.

**ObjClass**

The object class. The possible values are:

- Perm (permanent).
- Temp (temporary).

**DataSize**

The number of pages in the data object.

**LfSize** The number of pages in the long field object.

**LobSize**

The number of pages in the large object.

**XMLSize**

The number of pages in the XML object.

### TCB Table Stats:

**TableName**

The name of the table.

**Scans** The number of scans that have been performed against the table.

**UDI** The number of update, delete, and insert operations that have been performed against the table since the last time that the table statistics were updated through `RUNSTATS`.

**PgReorgs**

The number of page reorganizations performed.

**NoChgUpdts**

The number of updates that did not change any columns in the table.

## db2pd - Monitor and troubleshoot DB2 database

**Reads** The number of rows read from the table when the table switch was on for monitoring.

**FscrUpdates**

The number of updates to a free space control record.

**Inserts**

The number of insert operations performed on the table.

**Updates**

The number of update operations performed on the table.

**Deletes**

The number of delete operations performed on the table.

**OvFlReads**

The number of overflows read on the table when the table switch was on for monitoring.

**OvFlCrtes**

The number of new overflows that were created.

**Note** The following data is only displayed when the -all or -index option is specified with the -tcbstats parameter.

**TCB Index Information:**

**InxTbSpace**

The table space where the index resides.

**ObjectID**

The object identifier of the index.

**TbSpaceID**

The table space identifier.

**TableID**

The table identifier.

**MasterTbs**

For partitioned tables, this is the logical table space identifier to which the partitioned table belongs. For non-partitioned tables, this value corresponds to the TbSpaceID.

**MasterTab**

For partitioned tables, this is the logical table identifier of the partitioned table. For non-partitioned tables, this value corresponds to the TableID.

**TableName**

The name of the table.

**SchemaNm**

The schema that qualifies the table name.

**IID**

The index identifier.

**IndexObjSize**

The number of pages in the index object.

**TCB Index Stats:**

**TableName**

The name of the table.

**IID**

The index identifier.



**EmpPgDel**

The number of empty leaf nodes that were deleted.

**RootSplits**

The number of key insert or update operations that caused the index tree depth to increase.

**BndrySplits**

The number of boundary leaf splits that result in an insert operation into either the lowest or the highest key.

**PseuEmptPg**

The number of leaf nodes that are marked as being pseudo empty.

**Scans** The number of scans against the index.

**KeyUpdates**

The number of updates to the key.

**InclUpdats**

The number of included column updates.

**NonBndSpts**

The number of non-boundary leaf splits.

**PgAllocs**

The number of allocated pages.

**Merges**

The number merges performed on index pages.

**PseuDels**

The number of keys that are marked as pseudo deleted.

**DelClean**

The number of pseudo deleted keys that have been deleted.

**IntNodSpl**

The number of intermediate level splits.

**-reorg parameter:**

For the -reorg parameter, the following information is returned:

**Table Reorg Information:****TabSpaceID**

The table space identifier.

**TableID**

The table identifier.

**PartID**

The data partition identifier. One row is returned for each dataTpartition, showing the reorganization information.

**MasterTbs**

For partitioned tables, this is the logical table space identifier to which the partitioned table belongs. For non-partitioned tables, this value corresponds to the TbspaceID.

**MasterTab**

For partitioned tables, this is the logical table identifier of the partitioned table. For non-partitioned tables, this value corresponds to the TableID.

## db2pd - Monitor and troubleshoot DB2 database

### TableName

The name of the table.

**Type** The type of reorganization. The possible values are:

- Online
- Offline

### IndexID

The identifier of the index that is being used to reorganize the table.

### TempSpaceID

The table space in which the table is being reorganized.

### Table Reorg Stats:

#### TableName

The name of the table.

**Start** The time that the table reorganization started.

**End** The time that the table reorganization ended.

#### PhaseStart

The start time for a phase of table reorganization.

#### MaxPhase

The maximum number of reorganization phases that will occur during the reorganization. This value only applies to offline table reorganization.

**Phase** The phase of the table reorganization. This value only applies to offline table reorganization. The possible values are:

- Sort
- Build
- Replace
- InxRecreat

#### CurCount

A unit of progress that indicates the amount of table reorganization that has been completed. The amount of progress represented by this value is relative to the value of **MaxCount**, which indicates the total amount of work required to reorganize the table.

#### MaxCount

A value that indicates the total amount of work required to reorganize the table. This value can be used in conjunction with **CurCount** to determine the progress of the table reorganization.

**Status** The status of an online table reorganization. This value does not apply to offline table reorganizations. The possible values are:

- Started
- Paused
- Stopped
- Done
- Truncat

#### Completion

The success indicator for the table reorganization. The possible values are:

- 0. The table reorganization completed successfully.
- -1. The table reorganization failed.

### **-recovery parameter:**

For the -recovery parameter, the following information is returned:

#### **Recovery Status**

The internal recovery status.

#### **Current Log**

The current log being used by the recovery operation.

#### **Current LSN**

The current log sequence number.

#### **Job Type**

The type of recovery being performed. The possible values are:

- 5. Crash recovery.
- 6. Rollforward recovery on either the database or a table space.

#### **Job ID**

The job identifier.

#### **Job Start Time**

The time the recovery operation started.

#### **Job Description**

A description of the recovery activity. The possible values are:

- Tablespace Rollforward Recovery
- Database Rollforward Recovery
- Crash Recovery

#### **Invoker Type**

How the recovery operation was invoked. The possible values are:

- User
- DB2

#### **Total Phases**

The number of phases required to complete the recovery operation.

#### **Current phase**

The current phase of the recovery operation.

**Phase** The number of the current phase in the recovery operation.

#### **Forward phase**

The first phase of rollforward recovery. This phase is also known as the REDO phase.

#### **Backward phase**

The second phase of rollforward recovery. This phase is also known as the UNDO phase.

**Metric** The units of work. The possible values are:

- 1. Bytes.
- 2. Extents.
- 3. Rows.
- 4. Pages.
- 5. Indexes

#### **TotWkUnits**

The total number of units of work (UOW) to be done for this phase of the recovery operation.

## db2pd - Monitor and troubleshoot DB2 database

### **TotCompUnits**

The total number of UOWs that have been completed.

### **-reopt parameter:**

For the -reopt parameter, the following information is returned:

#### **Dynamic SQL Statements**

See “-dynamic” on page 203.

#### **Dynamic SQL Environments**

See the “-dynamic” on page 203.

#### **Dynamic SQL Variations**

See the “-dynamic” on page 203.

### **Reopt Values**

Displays information about the variables that were used to reoptimize a given SQL statement. Information is not returned for variables that were not used. Valid values are:

#### **AnchID**

The hash anchor identifier.

#### **StmtID**

The statement identifier for this variation.

**EnvID** The environment identifier for this variation.

**VarID** The variation identifier.

#### **OrderNum**

Ordinal number of the variable that was used to reoptimize of the SQL statement

#### **SQLZType**

The variable type.

#### **CodPg**

The variable code page.

**NulID** The flag indicating whether or not the value is null-terminated.

**Len** The length in bytes of the variable value.

**Data** The value used for the variable.

### **-osinfo parameter:**

For the -osinfo parameter, the following information is returned:

#### **CPU information: (On Windows, AIX, HP-UX, Solaris and Linux operating systems)**

##### **TotalCPU**

Total number of CPUs.

##### **OnlineCPU**

Number of CPUs online.

##### **ConfigCPU**

Number of CPUs configured.

##### **Speed(MHz)**

Speed, in MHz, of CPUs.

### **HMTDegree**

Systems supporting hardware multithreading return a value showing the number of processors that will appear to be present on the operating system. On nonHMT systems, this value is always 1. On HMT systems, TOTAL reflects the number of logical CPUs. To get the number of physical CPUs, divide the total by THREADING DEGREE.

### **Timebase**

Frequency, in Hz, of the timebase register increment. This is supported on Linux PPC only.

### **Physical memory and swap in megabytes: (On Windows, AIX, HP-UX, Solaris and Linux operating systems)**

#### **TotalMemTotal**

Size of memory in megabytes.

#### **FreeMem**

Amount of free memory in megabytes.

#### **AvailMem**

Amount of memory available to the product in megabytes.

#### **TotalSwap**

Total amount of swapspace in megabytes.

#### **FreeSwap**

Amount of swapspace free in megabytes.

### **Virtual memory in megabytes (On Windows, AIX, HP-UX, and Solaris operating systems)**

**Total** Total amount of virtual memory on the system in megabytes.

#### **Reserved**

Amount of reserved virtual memory in megabytes.

#### **Available**

Amount of virtual memory available in megabytes.

**Free** Amount of virtual memory free in megabytes.

### **Operating system information (On Windows, AIX, HP-UX, Solaris and Linux operating systems)**

#### **OSName**

Name of the operating system software.

#### **NodeName**

Name of the system.

#### **Version**

Version of the operating system.

#### **Machine**

Machine hardware identification.

### **Message queue information (On AIX, HP-UX, and Linux operating systems)**

#### **MsgSeg**

System-wide total of SysV msg segments.

#### **MsgMax**

System-wide maximum size of a message.

## db2pd - Monitor and troubleshoot DB2 database

### **MsgMap**

System-wide number of entries in message map.

### **MsgMni**

System-wide number of message queue identifiers for system.

### **MsgTql**

System-wide number of message headers.

### **MsgMnb**

Maximum number of bytes on a message queue.

### **MsgSsz**

Message segment size.

### **Shared memory information (On AIX, HP-UX, and Linux operating systems)**

#### **ShmMax**

System-wide maximum size of a shared memory segment in bytes.

#### **ShmMin**

System-wide minimum size of a shared memory segment in bytes.

#### **ShmIds**

System-wide number of shared memory identifiers.

#### **ShmSeg**

Process-wide maximum number of shared memory segments per process.

### **Semaphore information: (On AIX, HP-UX, and Linux operating systems)**

#### **SemMap**

System-wide number of entries in semaphore map.

#### **SemMni**

System-wide maximum number of a semaphore identifiers.

#### **SemMns**

System-wide maximum number of semaphores on system.

#### **SemMnu**

System-wide maximum number of undo structures on system.

#### **SemMsl**

System-wide maximum number of semaphores per ID.

#### **SemOpm**

System-wide maximum number of operations per semop call.

#### **SemUme**

Process-wide maximum number of undo structures per process.

#### **SemUsz**

System-wide size of undo structure. Derived from semume.

#### **SemVmx**

System-wide maximum value of a semaphore.

#### **SemAem**

System-wide maximum adjust on exit value.

### **CPU load information (On Windows, AIX, HP-UX, Solaris, and Linux operating systems)**

#### **shortPeriod**

The number of runnable processes over the preceeding 1 minute.

**mediumPeriod**

The number of runnable processes over the preceding 5 minutes.

**longPeriod**

The number of runnable processes over the preceding 15 minutes.

**Disk information****BkSz(bytes)**

File system block size in bytes.

**Total(bytes)**

Total number of bytes on the device in bytes.

**Free(bytes)**

Number of free bytes on the device in bytes.

**Inodes**

Total number of inodes.

**FSID** File system ID.

**DeviceType**

Device type.

**FSName**

File system name.

**MountPoint**

Mount point of the file system.

**-storagepaths parameter:**

For the -storagepaths parameter, the following information is returned:

**Number of Storage Paths**

The number of automatic storage paths defined for the database.

**PathName**

The name of an automatic storage path defined for the database.

**-pages parameter:**

For the -pages parameter, the following information is returned for each page:

**BPID** Bufferpool ID that contains the page.

**TbpaceID**

Table space ID that contains the page.

**TbpacePgNum**

Logical page number within the table space (DMS only).

**ObjID** Object ID that contains the page.

**ObjPgNum**

Logical page number within the object.

**ObjClass**

Class of object contained in the page. Possible values are Perm, Temp, Reorg, Shadow, and EMP.

**ObjType**

Type of object contained in the page. Possible values are Data, Index, LongField, XMLData, SMP, LOB, LOBA, and MDC\_BMP.

## db2pd - Monitor and troubleshoot DB2 database

**Dirty** Indicates if the page is dirty. Possible values are Y and N.

**Prefetched**

Indicates if the page has been prefetched. Possible values are Y and N.

**Related tasks:**

- “Identifying the owner of a lock that is being waited on” in *Troubleshooting Guide*

**Related reference:**

- “GET DATABASE CONFIGURATION ” on page 457
- “GET DATABASE MANAGER CONFIGURATION ” on page 463
- “db2pdcfg - Configure DB2 database for problem determination behavior” on page 223
- “SYSCAT.ROUTINES catalog view” in *SQL Reference, Volume 1*
- “SYSCAT.TABLES catalog view” in *SQL Reference, Volume 1*



## db2pdcfg - Configure DB2 database for problem determination behavior

Sets flags in the DB2 database memory sets to influence the database system behavior for problem determination purposes.

### Authorization:

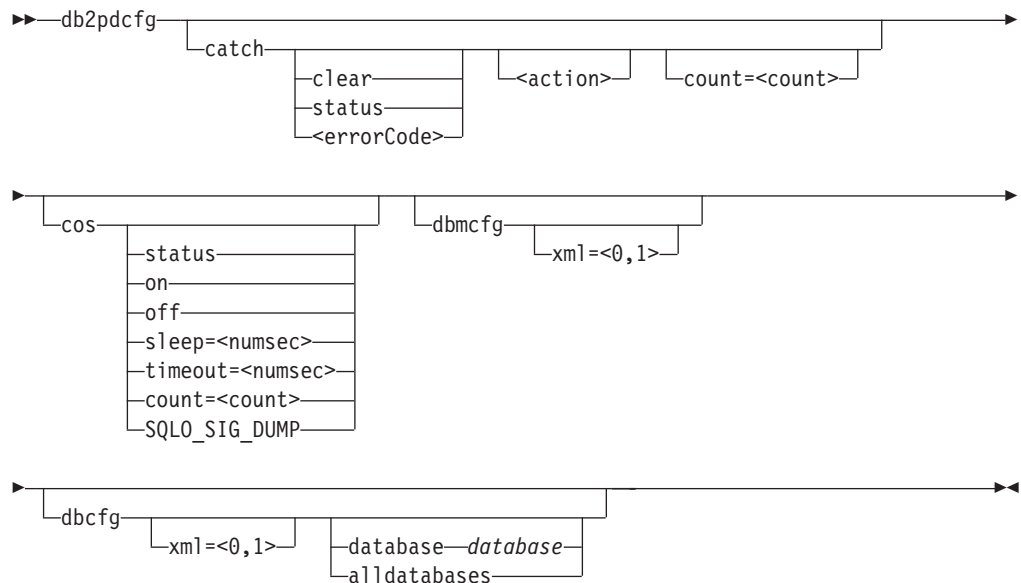
One of the following:

- On Linux and UNIX, the *sysadm* authority level. You must also be the instance owner.
- On Windows operating systems, the *sysadm* authority level.

### Required connection:

There is no minimum connection requirement. However, if a database scope option is specified, that database must be active before the command can return the requested information.

### Command syntax:



### Command parameters:

- catch** Instructs the database manager to catch an error or warning.
- Specify this option with the `clear` option to clear any catch flags that are set.
  - Specify this option with the `status` option to clear any catch flags that are set.
  - Specify this option with the `<errorCode>` option to clear any catch flags that are set.

Possible errorCodes are:

- `<sqlCode>[,<reasonCode>]`
- ZRC (hex or integer)
- ZRC #define (such as `SQLP_LTIMEOUT`)

## db2pdcfg

- ECF (hex or integer)
- "deadlock" or "locktimeout"
- Specify this option with the <action> option to set the desired action when the error or warning is caught by the database manager.

Possible actions are:

- [stack] (default) - Produce stack trace in db2diag.log
  - [db2cos] (default) - Run sqllib/db2cos callout script
  - [stopdb2trc] - Stop db2trc
  - [dumpcomponent] - Dump component flag
  - [component=<componentID>] - Component ID
  - [lockname=<lockname>] - Lockname for catching specific lock (lockname=000200030000001F0000000052)
  - [locktype=<locktype>] - Locktype for catching specific lock (locktype=R or locktype=52)
  - Specify this option with the count=<count> option to instruct the database manager the number of times to execute db2cos during a database manager trap. The default is 255.
- cos** Instructs the database manager how to invoke the db2cos callout script upon a database manager trap.
- Specify this option with the status option to print the status.
  - Specify this option with the off option to turn off the database manager call to db2cos during a database manager trap.
  - Specify this option with the on option to turn on the database manager call to db2cos during a database manager trap.
  - Specify this option with the sleep=<numsec> to instruct the database manager how long to sleep between checking the size of the output file generated by db2cos. The default is 3 seconds.
  - Specify this option with the timeout=<numsec> option to instruct the database manager how long of a timeout when checking if the output file generated by db2cos is growing in size. The default is 30 seconds.
  - Specify this option with the count=<count> option to instruct the database manager the number of times to execute db2cos during a database manager trap. The default is 255.
  - Specify this option with the SQL0\_SIG\_DUMP option to instruct the database manager to execute db2cos when receiving SQL0\_SIG\_DUMP signal.

### -dbmcfg

Sets DBM Config Reserved Bitmap to values 0 (default) or 1 (instance has xml data). This option is password protected which can be obtained from IBM DB2 Service.

### -dbcfg xml=<0,1>

Sets Database Config Reserved Bitmap to values 0 (default) or 1 (database has xml data). This option is password protected which can be obtained from IBM DB2 Service.

### Related concepts:

- "db2cos (callout script) output files" in *Troubleshooting Guide*

### Related reference:

- “db2pd - Monitor and troubleshoot DB2 database” on page 184

---

### db2perfc - Reset database performance values

Resets the performance values for one or more databases. It is used with the Performance Monitor on Windows operating systems.

#### Authorization:

Local Windows administrator.

#### Required connection:

None

#### Command syntax:



#### Command parameters:

- d** Specifies that performance values for DCS databases should be reset.
- dbalias* Specifies the databases for which the performance values should be reset. If no databases are specified, the performance values for all active databases will be reset..

#### Usage notes:

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.

The program resets the values for all programs currently accessing database performance information for the relevant DB2 server instance (that is, the one held in `db2instance` in the session in which you run **db2perfc**). Invoking **db2perfc** also resets the values seen by anyone remotely accessing DB2 performance information when the command is executed.

The `db2ResetMonitor` API allows an application to reset the values it sees locally, not globally, for particular databases.

#### Examples:

The following example resets performance values for all active DB2 databases:

```
db2perfc
```

The following example resets performance values for specific DB2 databases:

```
db2perfc dbalias1 dbalias2
```

The following example resets performance values for all active DB2 DCS databases:

```
db2perfc -d
```

The following example resets performance values for specific DB2 DCS databases:

## db2perfc - Reset Database Performance Values

```
db2perfc -d dbalias1 dbalias2
```

### Related reference:

- “db2ResetMonitor API - Reset the database system monitor data” in *Administrative API Reference*

---

### db2perfi - Performance counters registration utility

Adds the DB2 Performance Counters to the Windows operating system. This must be done to make DB2 and DB2 Connect performance information accessible to the Windows Performance Monitor.

#### Authorization:

Local Windows administrator.

#### Required connection:

None

#### Command syntax:

```
►► db2perfi [-i] [-u]
```

#### Command parameters:

- i Registers the DB2 performance counters.
- u Deregisters the DB2 performance counters.

#### Usage notes:

The **db2perfi -i** command will do the following:

1. Add the names and descriptions of the DB2 counter objects to the Windows registry.
2. Create a registry key in the Services key in the Windows registry as follows:

```
HKEY_LOCAL_MACHINE
 \System
 \CurrentControlSet
 \Services
 \DB2_NT_Performance
 \Performance
 Library=Name of the DB2 performance support DLL
 Open=Open function name, called when the DLL is
 first loaded
 Collect=Collect function name, called to request
 performance information
 Close=Close function name, called when the DLL is
 unloaded
```

#### Related tasks:

- “Registering DB2 with the Windows performance monitor” in *Administration Guide: Implementation*

#### Related reference:

- “db2perfc - Reset database performance values ” on page 226
- “db2perfr - Performance monitor registration tool ” on page 229

## db2perfr - Performance monitor registration tool

Used with the Performance Monitor on Windows operating systems. The db2perfr command is used to register an administrator user name and password with DB2 with DB2 when accessing the performance counters. This allows a remote Performance Monitor request to correctly identify itself to the DB2 database manager, and be allowed access to the relevant DB2 performance information. You also need to register an administrator user name and password if you want to log counter information into a file using the Performance Logs function.

### Authorization:

Local Windows administrator.

### Required connection:

None

### Command syntax:

```

▶▶ db2perfr [-r username password] [-u]

```

### Command parameters:

- r Registers the user name and password.
- u Deregisters the user name and password.

### Usage notes:

- Once a user name and password combination has been registered with DB2, even local instances of the Performance Monitor will explicitly log on using that user name and password. This means that if the user name information registered with DB2 does not match, local sessions of the Performance Monitor will not show DB2 performance information.
- The user name and password combination must be maintained to match the user name and password values stored in the Windows security database. If the user name or password is changed in the Windows security database, the user name and password combination used for remote performance monitoring must be reset.
- The default Windows Performance Monitor user name, SYSTEM, is a DB2 reserved word and cannot be used.

### Related tasks:

- “Enabling remote access to DB2 performance information” in *Administration Guide: Implementation*

### Related reference:

- “db2perfc - Reset database performance values ” on page 226
- “db2perfi - Performance counters registration utility ” on page 228

### db2rbind - Rebind all packages

Rebinds packages in a database.

#### Authorization:

One of the following:

- *sysadm*

#### Required connection:

None

#### Command syntax:

```
db2rbind database -l logfile [all] [-u userid -p password]
[-r [conservative] any]
```

#### 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.
- r Resolve. Specifies whether rebinding of the package is to be performed with or without conservative binding semantics. This affects whether new functions and data types are considered during function resolution and type resolution on static DML statements in the package. This option is not supported by DRDA. Valid values are:

##### conservative

Only functions and types in the SQL path that were defined before the last explicit bind time stamp are considered for function and type resolution. Conservative binding semantics are used. This is the default. This option is not supported for an inoperative package.

- any Any of the functions and types in the SQL path are considered for function and type resolution. Conservative binding semantics are not used.

#### Usage notes:



- This command uses the rebind API (sqlrbind) to attempt the revalidation of all packages in a database.
- Use of **db2rbind** is not mandatory.
- For packages that are invalid, you can choose to allow package revalidation to occur implicitly when the package is first used. You can choose to selectively revalidate packages with either the **REBIND** or the **BIND** command.
- If the rebind of any of the packages encounters a deadlock or a lock timeout the rebind of all the packages will be rolled back.

**Related reference:**

- “BIND” on page 355
- “PRECOMPILE ” on page 583
- “REBIND ” on page 619

## db2\_recon\_aid - Reconcile multiple tables

The db2\_recon\_aid utility provides an interface to the DB2 RECONCILE utility. The RECONCILE utility operates on one table at a time to validate all DATALINK column references in that table (and "repair" them accordingly). There are times when the RECONCILE utility might need to be run against multiple tables. db2\_recon\_aid is provided for this purpose.

Like the RECONCILE utility, the db2\_recon\_aid utility must be run on a DB2 server containing tables with DATALINK columns to be reconciled.

### Authorization:

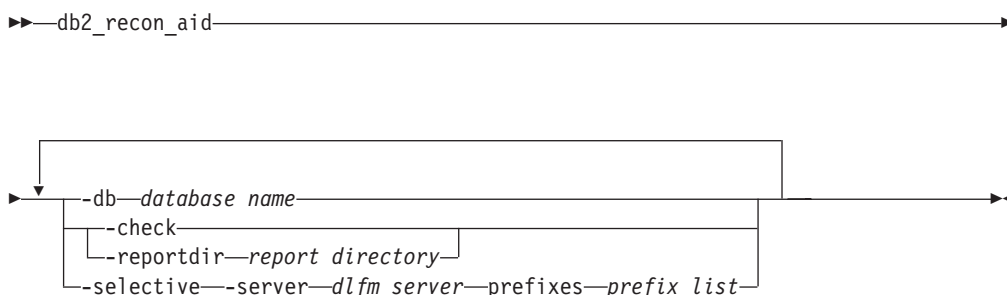
One of the following:

- sysadm
- sysctrl
- sysmaint
- dbadm

### Required connection:

None. This command automatically establishes a connection to the specified database.

### Command syntax:



Where *prefix list* is one or more DLFS prefixes delimited by a colon character, for instance *prefix1:prefix2:prefix3*.

### Command parameters:

#### -db *database name*

The name of the database containing the tables with DATALINK columns that need to be reconciled. This parameter is required.

**-check** List the tables that might need reconciliation. If you use this parameter, no reconcile operations will be performed. This parameter is required when the -reportdir parameter is not specified.

#### -reportdir

Specifies the directory where the utility is to place a report for each of the reconcile operations. For each table on which the reconcile is performed, files of the format <tbschema>.<tbname>.<ext> will be created where

- <tbschema> is the schema of the table;

## db2\_recon\_aid - RECONCILE Multiple Tables

- <tbname> is the table name;
- <ext> is .ulk or .exp. The .ulk file contains a list of files that were unlinked on the Data Links server, and the .exp file contains a list of files that were in exception on the Data Links server.

If -check and -reportdir are both specified, -reportdir is ignored.

### -selective

Process only those tables with DATALINK columns containing file references that match the specified -server and -prefixes criteria.

- If you use this parameter, you must also use both the -server and -prefixes parameters.
- If you do not use this parameter, then all Data Links servers and their prefixes that are registered with the specified DB2 database will either be reconciled, or will be flagged as needing reconciliation.

### -prefixes *prefix list*

Required when the -selective parameter is used. Specifies the name of one or more Data Links File System (DLFS) prefixes. Prefix values must start with a slash, and must be registered with the specified Data Links file server. Separate multiple prefix names with a colon(:), but do not include any embedded spaces. For example:

```
/dlfsdir1/smith:/dlfsdir2/smith/
```

The path in a DATALINK column value is considered to match the prefix list if any of the prefixes in the list are a left-most substring of the path.

If this parameter is not used, all prefixes for all Data Links servers that are registered with the specified DB2 database will be reconciled.

### -server

The name of the Data Links server for which the reconcile operation is to be performed. The parameter *dlfm server* represents an IP hostname. This hostname must exactly match the DLFM server hostname registered with the given DB2 database.

### Examples:

```
db2_recon_aid -db STAFF -check
```

```
db2_recon_aid -db STAFF -reportdir /home/smith
```

```
db2_recon_aid -db STAFF -check -selective -server dlmsvr.services.com
-prefixes /dlfsdir1/smith/
```

```
db2_recon_aid -db STAFF -reportdir /home/smith -selective -server
dlmsvr.services.com -prefixes /dlfsdir1/smith:/dlfsdir2/smith/
```

### Usage notes:

1. On AIX systems or Solaris Operating Environments, the db2\_recon\_aid utility is located in the INSTHOME/sqllib/adm directory, where INSTHOME is the home directory of the instance owner.
2. On Windows systems, the utility is located in x:\sqllib\bin directory where x: is the drive where you installed DB2 Data Links Manager.
3. db2\_recon\_aid can identify all tables in a given database which contain DATALINK columns with the FILE LINK CONTROL column attribute. It is these types of columns which might require file reference validation via the RECONCILE utility. By specifying the -check option, the tables of interest can

## db2\_recon\_aid - RECONCILE Multiple Tables

be simply listed. By specifying the `-reportdir` option, the RECONCILE utility can actually be automatically run against this set of tables. By specifying the `-selective` option, you can narrow down the set of tables which `db2_recon_aid` identifies as candidates for reconciliation (based upon the table's DATALINK column(s) containing references to a specific Data Links server and one or more of its Data Links File Systems).

4. Depending upon what problem you are trying to solve, you will need to choose between running the RECONCILE or `db2_recon_aid` utility. The overriding consideration is how many tables might need to be reconciled. For example:
  - If you have an individual table in a state like DRP or DRNP, you might only need to run RECONCILE for that specific table to restore the table to a normal state.
  - If you have had a corruption or loss of a Data Links File System (DLFS) on a given Data Links server, you should use `db2_recon_aid` (with the `-selective` option) to locate all tables referencing that Data Links server and that specific "prefix" (DLFS path), and perform the reconciliation on each of these tables.
  - If you are simply wanting to validate ALL of your DATALINK file references in your database, you would run `db2_recon_aid` (without the `-selective` option).
5. Each prefix must be an absolute path (the path must start with a slash), and must be registered with the given DLFM server.
6. The path in a DATALINK column value is considered to match the prefix list if any of the prefixes in the list are a leftmost substring of the path.

### Related reference:

- "RECONCILE " on page 623

## db2relocatedb - Relocate database

This command renames a database, or relocates a database or part of a database (for example, the container and the log directory) as specified in the configuration file provided by the user. This tool makes the necessary changes to the DB2 instance and database support files.

### Authorization:

None

### Command syntax:

```
▶▶—db2relocatedb—-f—configFilename—▶▶
```

### Command parameters:

#### -f configFilename

Specifies the name of the file containing the configuration information necessary for relocating the database. This can be a relative or absolute file name. The format of the configuration file is:

```
DB_NAME=oldName,newName
DB_PATH=oldPath,newPath
INSTANCE=oldInst,newInst
NODENUM=nodeNumber
LOG_DIR=oldDirPath,newDirPath
CONT_PATH=oldContPath1,newContPath1
CONT_PATH=oldContPath2,newContPath2
...
STORAGE_PATH=oldStoragePath1,newStoragePath1
STORAGE_PATH=oldStoragePath2,newStoragePath2
...
```

Where:

#### DB\_NAME

Specifies the name of the database being relocated. If the database name is being changed, both the old name and the new name must be specified. This is a required field.

#### DB\_PATH

Specifies the original path of the database being relocated. If the database path is changing, both the old path and new path must be specified. This is a required field.

#### INSTANCE

Specifies the instance where the database exists. If the database is being moved to a new instance, both the old instance and new instance must be specified. This is a required field.

#### NODENUM

Specifies the node number for the database node being changed. The default is 0.

#### LOG\_DIR

Specifies a change in the location of the log path. If the log path is being changed, both the old path and new path must be specified. This specification is optional if the log path resides under the database path, in which case the path is updated automatically.

## db2relocatedb - Relocate Database

### CONT\_PATH

Specifies a change in the location of table space containers. Both the old and new container path must be specified. Multiple CONT\_PATH lines can be provided if there are multiple container path changes to be made. This specification is optional if the container paths reside under the database path, in which case the paths are updated automatically. If you are making changes to more than one container where the same old path is being replaced by a common new path, a single CONT\_PATH entry can be used. In such a case, an asterisk (\*) could be used both in the old and new paths as a wildcard.

### STORAGE\_PATH

This is only applicable to databases with automatic storage enabled. It specifies a change in the location of one of the storage paths for the database. Both the old storage path and the new storage path must be specified. Multiple STORAGE\_PATH lines can be given if there are several storage path changes to be made.

Blank lines or lines beginning with a comment character (#) are ignored.

### Usage notes:

If the instance that a database belongs to is changing, the following must be done before running this command to ensure that changes to the instance and database support files are made:

- If a database is being moved to another instance, create the new instance.
- Copy the files and devices belonging to the databases being copied onto the system where the new instance resides. The path names must be changed as necessary. However, if there are already databases in the directory where the database files are moved to, you can mistakenly overwrite the existing sqlbdir file, thereby removing the references to the existing databases. In this scenario, the **db2relocatedb** utility cannot be used. Instead of **db2relocatedb**, an alternative is a redirected restore operation.
- Change the permission of the files/devices that were copied so that they are owned by the instance owner.

If the instance is changing, the tool must be run by the new instance owner.

In a partitioned database environment, this tool must be run against every database partition that requires changes. A separate configuration file must be supplied for each database partition, that includes the NODENUM value of the database partition being changed. For example, if the name of a database is being changed, every database partition will be affected and the **db2relocatedb** command must be run with a separate configuration file on each database partition. If containers belonging to a single database partition are being moved, the **db2relocatedb** command only needs to be run once on that database partition.

### Examples:

#### Example 1

To change the name of the database TESTDB to PRODDB in the instance db2inst1 that resides on the path /home/db2inst1, create the following configuration file:

```
DB_NAME=TESTDB,PRODDB
DB_PATH=/home/db2inst1
INSTANCE=db2inst1
NODENUM=0
```

Save the configuration file as `relocate.cfg` and use the following command to make the changes to the database files:

```
db2relocatedb -f relocate.cfg
```

### Example 2

To move the database `DATAB1` from the instance `jsmith` on the path `/dbpath` to the instance `prodinst` do the following:

1. Move the files in the directory `/dbpath/jsmith` to `/dbpath/prodinst`.
2. Use the following configuration file with the **db2relocatedb** command to make the changes to the database files:

```
DB_NAME=DATAB1
DB_PATH=/dbpath
INSTANCE=jsmith,prodinst
NODENUM=0
```

### Example 3

The database `PRODDB` exists in the instance `inst1` on the path `/databases/PRODDB`. The location of two table space containers needs to be changed as follows:

- SMS container `/data/SMS1` needs to be moved to `/DATA/NewSMS1`.
- DMS container `/data/DMS1` needs to be moved to `/DATA/DMS1`.

After the physical directories and files have been moved to the new locations, the following configuration file can be used with the **db2relocatedb** command to make changes to the database files so that they recognize the new locations:

```
DB_NAME=PRODDB
DB_PATH=/databases/PRODDB
INSTANCE=inst1
NODENUM=0
CONT_PATH=/data/SMS1,/DATA/NewSMS1
CONT_PATH=/data/DMS1,/DATA/DMS1
```

### Example 4

The database `TESTDB` exists in the instance `db2inst1` and was created on the path `/databases/TESTDB`. Table spaces were then created with the following containers:

```
TS1
TS2_Cont0
TS2_Cont1
/databases/TESTDB/TS3_Cont0
/databases/TESTDB/TS4/Cont0
/Data/TS5_Cont0
/dev/rTS5_Cont1
```

`TESTDB` is to be moved to a new system. The instance on the new system will be `newinst` and the location of the database will be `/DB2`.

When moving the database, all of the files that exist in the `/databases/TESTDB/db2inst1` directory must be moved to the `/DB2/newinst` directory. This means that the first 5 containers will be relocated as part of this move. (The first 3 are relative

## db2relocatedb - Relocate Database

to the database directory and the next 2 are relative to the database path.) Since these containers are located within the database directory or database path, they do not need to be listed in the configuration file. If the 2 remaining containers are to be moved to different locations on the new system, they must be listed in the configuration file.

After the physical directories and files have been moved to their new locations, the following configuration file can be used with **db2relocatedb** to make changes to the database files so that they recognize the new locations:

```
DB_NAME=TESTDB
DB_PATH=/databases/TESTDB,/DB2
INSTANCE=db2inst1,newinst
NODENUM=0
CONT_PATH=/Data/TS5_Cont0,/DB2/TESTDB/TS5_Cont0
CONT_PATH=/dev/rTS5_Cont1,/dev/rTESTDB_TS5_Cont1
```

### Example 5

The database TESTDB has two database partitions on database partition servers 10 and 20. The instance is servinst and the database path is /home/servinst on both database partition servers. The name of the database is being changed to SERVDB and the database path is being changed to /databases on both database partition servers. In addition, the log directory is being changed on database partition server 20 from /testdb\_logdir to /servdb\_logdir.

Since changes are being made to both database partitions, a configuration file must be created for each database partition and **db2relocatedb** must be run on each database partition server with the corresponding configuration file.

On database partition server 10, the following configuration file will be used:

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=10
```

On database partition server 20, the following configuration file will be used:

```
DB_NAME=TESTDB,SERVDB
DB_PATH=/home/servinst,/databases
INSTANCE=servinst
NODE_NUM=20
LOG_DIR=/testdb_logdir,/servdb_logdir
```

### Example 6

The database MAINDB exists in the instance maininst on the path /home/maininst. The location of four table space containers needs to be changed as follows:

```
/maininst_files/allconts/C0 needs to be moved to /MAINDB/C0
/maininst_files/allconts/C1 needs to be moved to /MAINDB/C1
/maininst_files/allconts/C2 needs to be moved to /MAINDB/C2
/maininst_files/allconts/C3 needs to be moved to /MAINDB/C3
```

After the physical directories and files are moved to the new locations, the following configuration file can be used with the **db2relocatedb** command to make changes to the database files so that they recognize the new locations.



A similar change is being made to all of the containers; that is, /maininst\_files/allconts/ is being replaced by /MAINDB/ so that a single entry with the wildcard character can be used:

```
DB_NAME=MAINDB
DB_PATH=/home/maininst
INSTANCE=maininst
NODE_NUM=0
CONT_PATH=/maininst_files/allconts/*, /MAINDB/*
```

### Related reference:

- “db2inidb - Initialize a mirrored database ” on page 130

### db2rfpen - Reset rollforward pending state

Puts a database in rollforward pending state. If you are using high availability disaster recovery (HADR), the database is reset to a standard database.

**Authorization:**

None

**Required connection:**

None

**Command syntax:**

```
▶▶—db2rfpen—ON—database_alias—-log—logfile_path—————▶▶
```

**Command parameters:**

**database\_alias**

Specifies the name of the database to be placed in rollforward pending state. If you are using high availability disaster recovery (HADR), the database is reset to a standard database.

**-log logfile\_path**

Specifies the log file path.

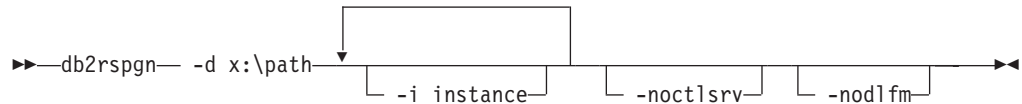
**Related concepts:**

- “High availability disaster recovery overview” in *Data Recovery and High Availability Guide and Reference*

## db2rspgn - Response file generator (Windows)

The **db2rspgn** command is available only on Windows.

### Command syntax:



### Command parameters:

- d** Destination directory for a response file and any instance files. This parameter is required.
- i** A list of instances for which you want to create a profile. The default is to generate an instance profile file for all instances. This parameter is optional.
- noctlsrv** Indicates that an instance profile file will not be generated for the Control Server instance. This parameter is optional.
- nodlfm** Indicates that an instance profile file will not be generated for the Data Links File Manager instance. This parameter is optional.

### Related concepts:

- “The response file generator (Windows)” in *Installation and Configuration Supplement*

### Related tasks:

- “Response file installation of DB2 overview (Windows)” in *Installation and Configuration Supplement*

### db2sampl - Create sample database

Creates a sample database named SAMPLE.

This database will not be automatically configured when it is first created. Users can issue the **AUTOCONFIGURE** command against the SAMPLE database at a later time.

#### Authorization:

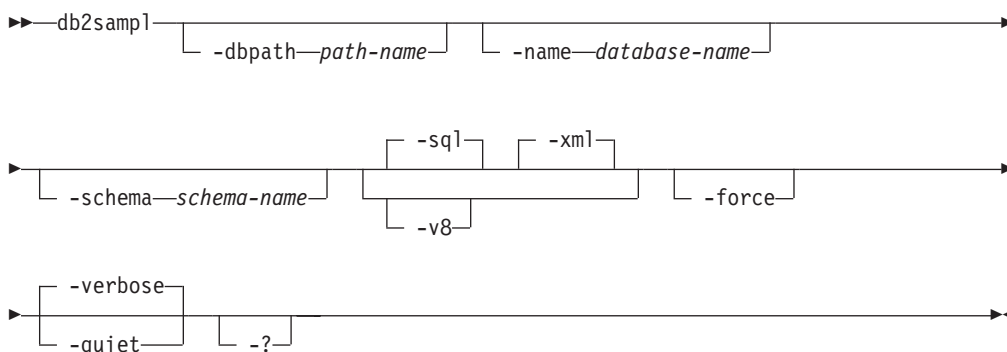
One of the following:

- *sysadm*
- *sysctrl*

#### Required Connection:

None

#### Command syntax:



#### Command parameters:

##### **-dbpath** *path-name*

Specifies the path on which to create the database. On Windows operating systems, specifies the letter of the drive on which to create the database. The maximum length for *path-name* is 175 characters. By default, *path-name* is the default path specified in the database manager configuration file (`dftdbpath` parameter).

##### **-name** *database-name*

Specifies a name for the sample database. The database name must adhere to the naming conventions for databases. By default, *database-name* is SAMPLE.

##### **-schema** *schema-name*

Specifies the default schema in which to create the database objects. The names of all database objects will be qualified with the schema name. The schema name must adhere to the naming conventions for schemas. By default, *schema-name* is the value of the `CURRENT_SCHEMA` special register which corresponds to the current user's authorization ID.

**-sql** Creates tables, triggers, functions, procedures, and populates the tables with data.

**-xml** Creates tables with columns of data type XML, creates indexes on the XML columns, registers XML schemas, and populates these tables with data including XML document values.

When this option is specified either explicitly or implied by default, the sample database is created with a Unicode (UTF-8) codeset which is a prerequisite for working with native XML features. It is created with a UCA400\_NO collation, and a C (Canadian) territory.

This option is only supported where XML is supported. If XML is not supported, this option is ignored.

**-v8** Creates the DB2 Universal Database Version 8 sample database, database objects and data. The Version 8 sample database is a non-unicode database named SAMPLE that is created in the default path specified in the database manager configuration file (dftdbpath parameter).

**-force** Forces the drop and recreation of any existing database in the instance with the same name as specified for the sample database.

**-verbose**

Prints status messages to standard output.

**-quiet** Suppresses the printing of status messages to standard output.

**-?** Returns the **db2sampl** command syntax help.

### Default behaviour of db2sampl

When the **db2sampl** command is issued without any optional arguments, depending on whether the environment is partitioned or not, it behaves differently:

In non-partitioned database environments:

- Creates a database named SAMPLE with a Unicode (UTF-8) codeset and a UCA400\_NO collation and a C (Canadian) territory in the default database path.
- Creates relational database objects including tables, indexes, constraints, triggers, functions, procedures, multi-dimensional clustered tables and materialized query tables.
- Populates relational tables with data.
- Creates tables with XML data type columns.
- Creates indexes over XML data.
- Creates an XML schema repository that contains XML schema documents.
- All database object names are qualified with the value of the CURRENT\_SCHEMA special register.

In partitioned database environments:

- Creates a database named SAMPLE with the default codeset and collation derived from the operating system environment.
- Creates relational database objects including tables, indexes, constraints, triggers, functions, procedures, multi-dimensional clustered tables and materialized query tables.
- Populates tables with data.
- All database object names are qualified with the value of the CURRENT\_SCHEMA special register.

**Usage notes:**

## db2sampl - Create Sample Database

- The **db2sampl** command can only be issued on a computer where a DB2 database server is installed. It cannot be issued from a remote DB2 client.
- Use of the **db2sampl** command to create sample databases with XML database objects will prohibit future use of the Data Partitioning Feature available with DB2 Enterprise Server Edition. Warning text is sent to standard output if the **db2sampl** command is issued and DB2 Enterprise Server Edition is installed.
- The sample database is created with the instance authentication type that is specified by the database manager configuration parameter, authentication.

### Examples:

- To create a sample database with the default characteristics, issue:  
`db2sampl`
- On Windows operating systems, to create a sample database named *mysample* on the *E:* drive containing only SQL database objects in schema *myschema* and to view status messages, issue:  
`db2sampl -dbpath E -name mysample -schema myschema -sql -force -verbose`
- To create the DB2 Version 8 sample database, issue:  
`db2sampl -v8`

### Related tasks:

- “Creating the sample database” in *Samples Topics*

### Related reference:

- “The SAMPLE database” in *Samples Topics*
- “GET DATABASE MANAGER CONFIGURATION ” on page 463
- Appendix B, “Naming conventions,” on page 797
- “CREATE DATABASE ” on page 395

## db2set - DB2 profile registry

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.

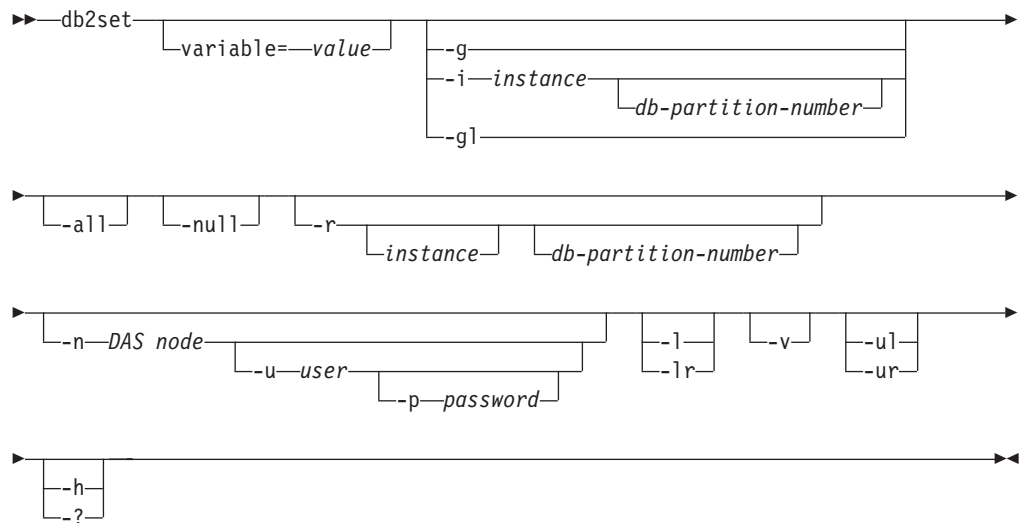
### Authorization:

*sysadm*

### Required connection:

None

### Command syntax:



### 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 registry variables for all instances pertaining to a particular DB2 copy.

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

#### db-partition-number

Specifies a number listed in the `db2nodes.cfg` file.

**-gl** Accesses the global profile variables stored in LDAP. This option is only effective if the registry variable `DB2_ENABLE_LDAP` has been set to YES.

**-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].

## db2set - DB2 Profile Registry Command

- 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. If no instance is specified, and an instance attachment exists, resets the profile for the current instance. If no instance is specified, and no attachment exists, resets the profile for the instance specified by the DB2INSTANCE environment variable.
- 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 for the current DB2 product installation.
- lr** Lists all supported registry variables.
- v** Specifies verbose mode.
- ul** Accesses the user profile variables. This parameter is supported on Windows operating systems only.
- ur** Refreshes the user profile variables. This parameter is supported on Windows operating systems only.
- 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) pertaining to a particular installation:  
:  
`db2set -l`
- Display all supported registry variables:  
`db2set -lr`
- Display all defined global variables which are visible by all instances pertaining to a particular installation:  
`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 globally for all instances pertaining to a particular installation:



## db2set - DB2 Profile Registry Command

```
db2set -g DB2COMM=TCPIP
```

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

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

### Related reference:

- “REG\_VARIABLES administrative view – Retrieve DB2 registry settings in use” in *Administrative SQL Routines and Views*

## db2setup - Install DB2

Installs DB2 products. This command is only available on Linux and UNIX-based systems. The command for Windows operating systems is **setup**.

This utility is located on the DB2 installation media. It launches the DB2 Setup wizard to define the installation and install DB2 products. If invoked with the *-r* option, it performs an installation without further input, taking installation configuration information from a response file.

### Authorization:

Root access on Linux and UNIX-based systems.

### Command syntax:

```

▶▶ db2setup [-i language] [-l log_file] [-t trace_file]
▶ [-r response_file] [--?] [-h]

```

### Command parameters:

- i** *language*  
Two-letter language code of the language in which to perform the installation.
- l** *log\_file*  
Full path and file name of the log file to use.
- t** *trace\_file*  
Generates a file with install trace information.
- r** *response\_file*  
Full path and file name of the response file to use.
- , -h** Generates usage information.

### Usage Notes:

You must log on as root or use `su` with the `-` flag to set the process environment as if you had logged in as root. If the process environment is not set as root, the installation process finishes without errors but you will encounter errors when you run the DB2 copy.

### Related reference:

- “setup - Install DB2” on page 305
- “Language identifiers for running the DB2 Setup wizard in another language” in *Quick Beginnings for DB2 Servers*

## 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.

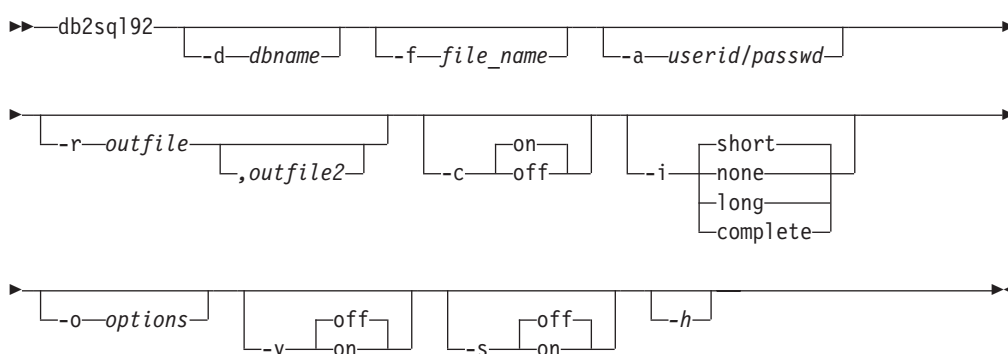
### Authorization:

*sysadm*

### Required connection:

None. This command establishes a database connection.

### Command syntax:



### 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).

## db2sql92 - SQL92 Compliant SQL Statement Processor

### 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

## db2sql92 - SQL92 Compliant SQL Statement Processor

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 65 535 characters in length. Statements must be terminated by a semicolon.

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

When running queries, there is no support for the results set to include LOBs.

### Related reference:

- “db2batch - Benchmark tool ” on page 34

## db2sqljbind - SQLJ profile binder

db2sqljbind binds DB2 packages for a serialized profile that was previously customized with the db2sqljcustomize command.

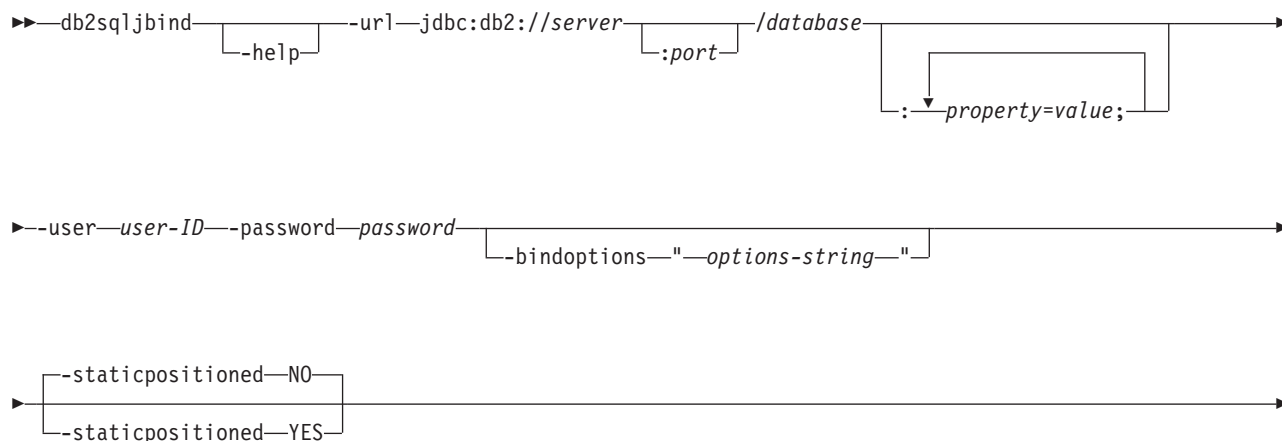
### Authorization:

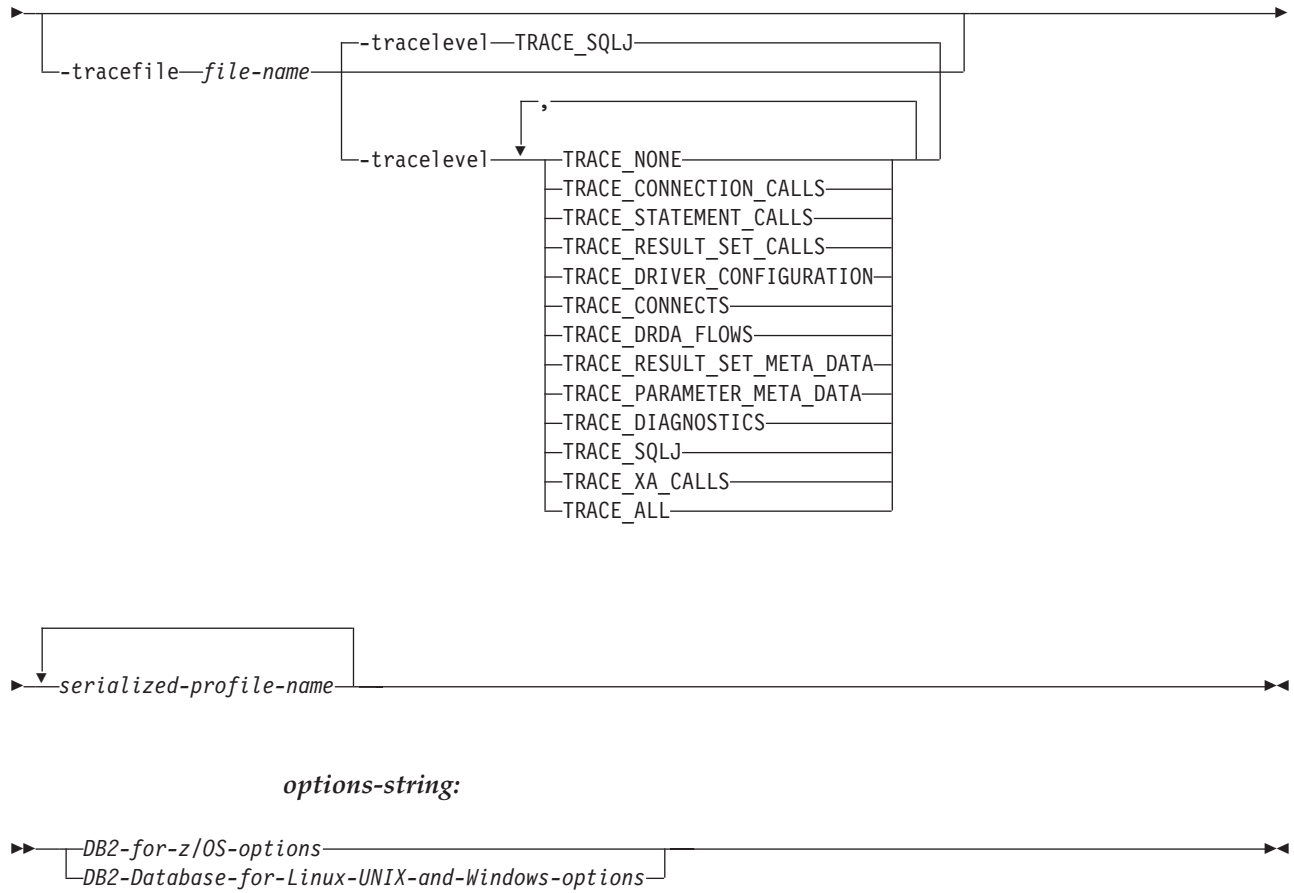
The privilege set of the process must include one of the following authorities:

- SYSADM authority
- DBADM authority
- If the package does not exist, the BINDADD privilege, and one of the following privileges:
  - CREATEIN privilege
  - IMPLICIT\_SCHEMA authority on the database if the schema name of the package does not exist
- If the package exists:
  - ALTERIN privilege on the schema
  - BIND privilege on the package

The user also needs all privileges that are required to compile any static SQL statements in the application. Privileges that are granted to groups are not used for authorization checking of static statements. If the user has SYSADM authority, but no explicit privileges to complete the bind, the DB2 database manager grants explicit DBADM authority automatically.

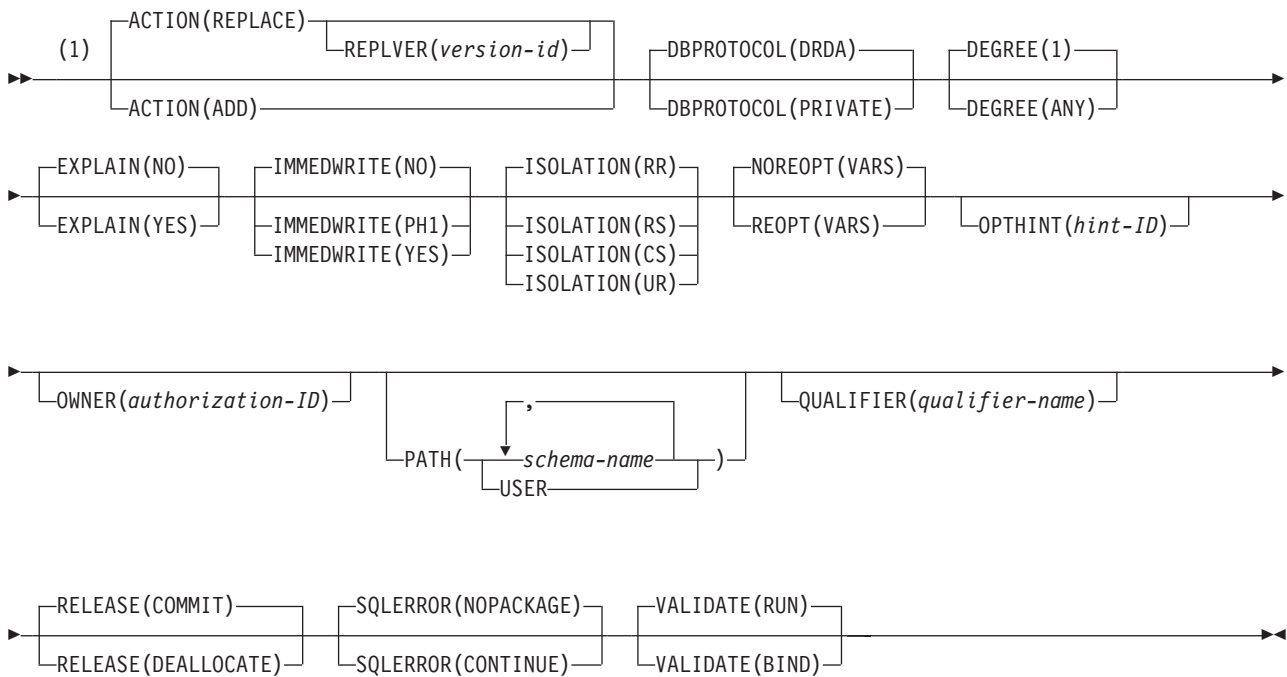
### Command syntax:





## db2sqljbind - SQLJ profile binder

### DB2 for z/OS options:

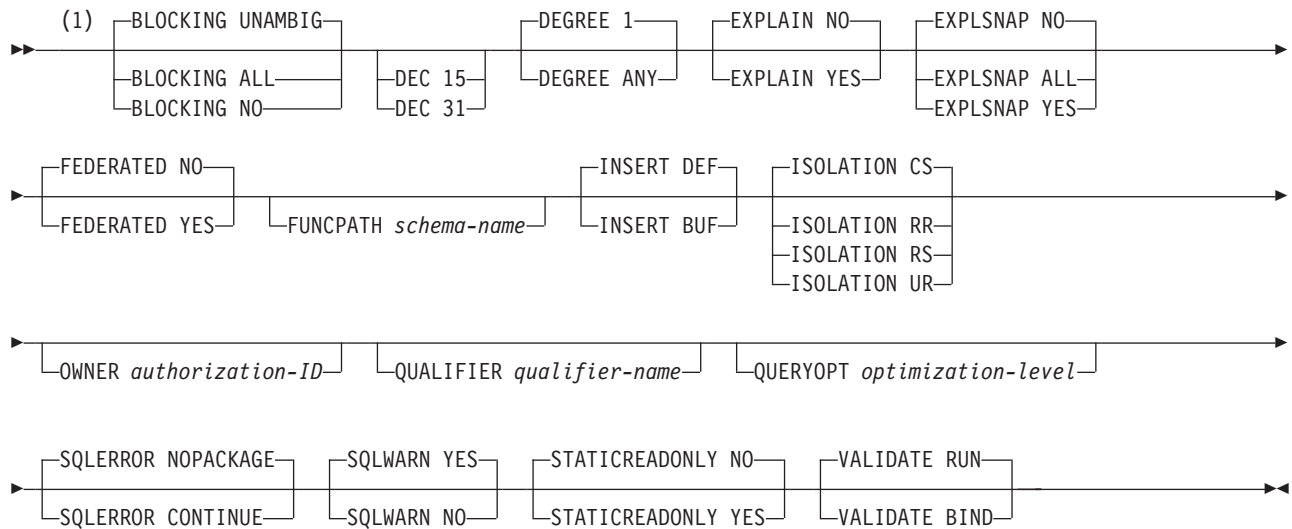


### Notes:

- 1 These options can be specified in any order.



## DB2 Database for Linux, UNIX, and Windows options

**Notes:**

- 1 These options can be specified in any order.

**Command parameters:****-help**

Specifies that db2sqljbind describes each of the options that it supports. If any other options are specified with -help, they are ignored.

**-url**

Specifies the URL for the data source for which the profile is to be customized. This URL is used if the -automaticbind or -onlinecheck option is YES. The variable parts of the -url value are:

**server**

The domain name or IP address of the MVS system on which the DB2 subsystem resides.

**port**

The TCP/IP server port number that is assigned to the DB2 subsystem. The default is 446.

**database**

A name for the database server for which the profile is to be customized.

If the connection is to a DB2 for z/OS server, *database* is the DB2 location name that is defined during installation. All characters in this value must be uppercase characters. You can determine the location name by executing the following SQL statement on the server:

```
SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1;
```

If the connection is to a DB2 Database for Linux, UNIX, and Windows server, *database* is the database name that is defined during installation.

If the connection is to an IBM Cloudscape server, the *database* is the fully-qualified name of the file that contains the database. This name must be enclosed in double quotation marks ("). For example:

```
"c:/databases/testdb"
```

## db2sqljbind - SQLJ profile binder

*property=value;*

A property for the JDBC connection. For the definitions of these properties, see Properties for the IBM DB2 Driver for JDBC and SQLJ.

**-user** *user-ID*

Specifies the user ID to be used to connect to the data source for binding the package.

**-password** *password*

Specifies the password to be used to connect to the data source for binding the package.

**-bindoptions** *options-string*

Specifies a list of options, separated by spaces. These options have the same function as DB2 precompile and bind options with the same names. If you are preparing your program to run on a DB2 for z/OS system, specify DB2 for z/OS options. If you are preparing your program to run on a DB2 Database for Linux, UNIX, and Windows system, specify DB2 Database for Linux, UNIX, and Windows options.

### **Notes on bind options:**

- Specify VERSION only if the following conditions are true:
  - If you are binding a package at a DB2 Database for Linux, UNIX, and Windows system, the system is at Version 8 or later.
  - You rerun the translator on a program before you bind the associated package with a new VERSION value.
- The value for STATICREADONLY is YES for servers that support STATICREADONLY, and NO for other servers. When you specify STATICREADONLY YES, DB2 processes ambiguous cursors as if they were read-only cursors. For troubleshooting iterator declaration errors, you need to explicitly specify STATICREADONLY NO, or declare iterators so that they are unambiguous. For example, if you want an iterator to be unambiguously updatable, declare the iterator to implement sqlj.runtime.ForUpdate. If you want an iterator to be read-only, include the FOR READ ONLY clause in SELECT statements that use the iterator.

**Important:** Specify only those program preparation options that are appropriate for the data source at which you are binding a package. Some values and defaults for the IBM DB2 Driver for JDBC and SQLJ are different from the values and defaults for DB2.

**-staticpositioned** **NO|YES**

For iterators that are declared in the same source file as positioned UPDATE statements that use the iterators, specifies whether the positioned UPDATES are executed as statically bound statements. The default is NO. NO means that the positioned UPDATES are executed as dynamically prepared statements. This value must be the same as the -staticpositioned value for the previous db2sqljcustomize invocation for the serialized profile.

**-tracefile** *file-name*

Enables tracing and identifies the output file for trace information. This option should be specified only under the direction of IBM Software Support.

**-tracelevel**

If -tracefile is specified, indicates what to trace while db2sqljcustomize runs. The default is TRACE\_SQLJ. This option should be specified only under the direction of IBM Software Support.

*serialized-profile-name*

Specifies the name of one or more serialized profiles from which the package is bound. A serialized profile name is of the following form:

```
program-name_SJProfileIDNumber.ser
```

*program-name* is the name of the SQLJ source program, without the extension .sqlj. *n* is an integer between 0 and *m*-1, where *m* is the number of serialized profiles that the SQLJ translator generated from the SQLJ source program.

If you specify more than one serialized profile name to bind a single DB2 package from several serialized profiles, you must have specified the same serialized profile names, in the same order, when you ran db2sqljcustomize.

**Examples:**

```
db2sqljbind -user richler -password mordecai
 -url jdbc:db2://server:50000/sample -bindoptions "EXPLAIN YES"
 pgmname_SJProfile0.ser
```

**Usage notes:**

**Package names produced by db2sqljbind:** The names of the packages that are created by db2sqljbind are the names that you specified using the-rootpkgname or -singlepkgname parameter when you ran db2sqljcustomize. If you did not specify -rootpkgname or -singlepkgname, the package names are the first seven bytes of the profile name, appended with the isolation level character.

**DYNAMICRULES value for db2sqljbind:** The DYNAMICRULES bind option determines a number of run-time attributes for a DB2 package. Two of those attributes are the authorization ID that is used to check authorization, and the qualifier that is used for unqualified objects. To ensure the correct authorization for dynamically executed positioned UPDATE and DELETE statements in SQLJ programs, db2sqljbind always binds the DB2 packages with the DYNAMICRULES(BIND) option. You cannot modify this option. The DYNAMICRULES(BIND) option causes the SET CURRENT SQLID statement and the SET CURRENT SCHEMA statement to have no impact on an SQLJ program, because those statements affect only dynamic statements that are bound with DYNAMICRULES values other than BIND.

With DYNAMICRULES(BIND), unqualified table, view, index, and alias names in dynamic SQL statements are implicitly qualified with value of the bind option QUALIFIER. If you do not specify QUALIFIER, DB2 uses the authorization ID of the package owner as the implicit qualifier. If this behavior is not suitable for your program, you can use one of the following techniques to set the correct qualifier:

- Force positioned UPDATE and DELETE statements to execute statically. You can use the -staticpositioned YES option of db2sqljcustomize or db2sqljbind to do this if the cursor (iterator) for a positioned UPDATE or DELETE statement is in the same package as the positioned UPDATE or DELETE statement.
- Fully qualify DB2 table names in positioned UPDATE and positioned DELETE statements.

**Related reference:**

- “BIND” on page 355
- “db2sqljcustomize - SQLJ profile customizer” on page 259
- “db2sqljprint - SQLJ profile printer” on page 270

## db2sqljbind - SQLJ profile binder

- “sqlj - SQLJ translator” on page 307

## db2sqljcustomize - SQLJ profile customizer

db2sqljcustomize processes an SQLJ profile, which contains embedded SQL statements. By default, db2sqljcustomize produces four DB2 packages: one for each isolation level. db2sqljcustomize augments the profile with DB2-specific information for use at run time.

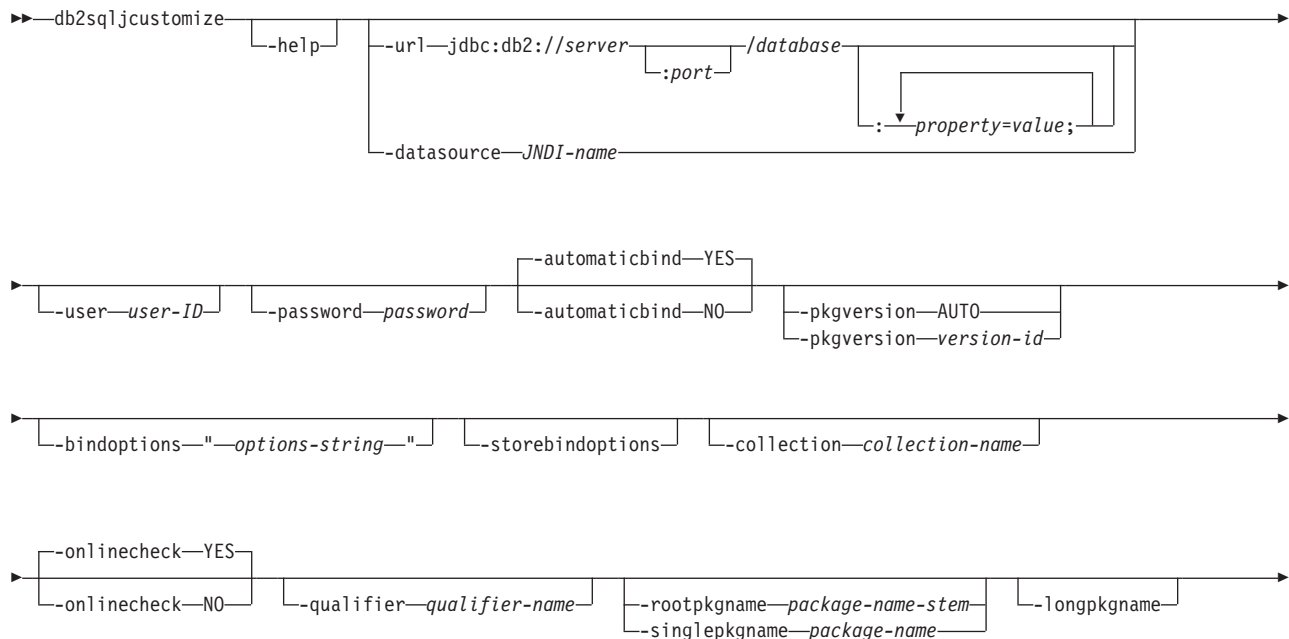
### Authorization:

The privilege set of the process must include one of the following authorities:

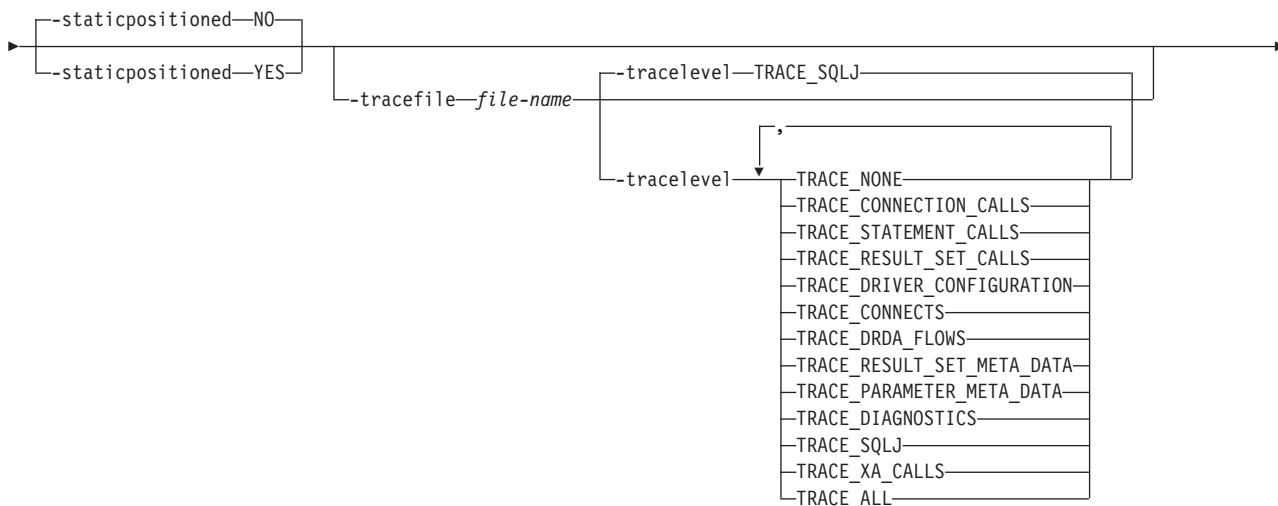
- SYSADM authority
- DBADM authority
- If the package does not exist, the BINDADD privilege, and one of the following privileges:
  - CREATEIN privilege
  - IMPLICIT\_SCHEMA authority on the database if the schema name of the package does not exist
- If the package exists:
  - ALTERIN privilege on the schema
  - BIND privilege on the package

The user also needs all privileges that are required to compile any static SQL statements in the application. Privileges that are granted to groups are not used for authorization checking of static statements. If the user has SYSADM authority, but no explicit privileges to complete the bind, the DB2 database manager grants explicit DBADM authority automatically.

### Command syntax:



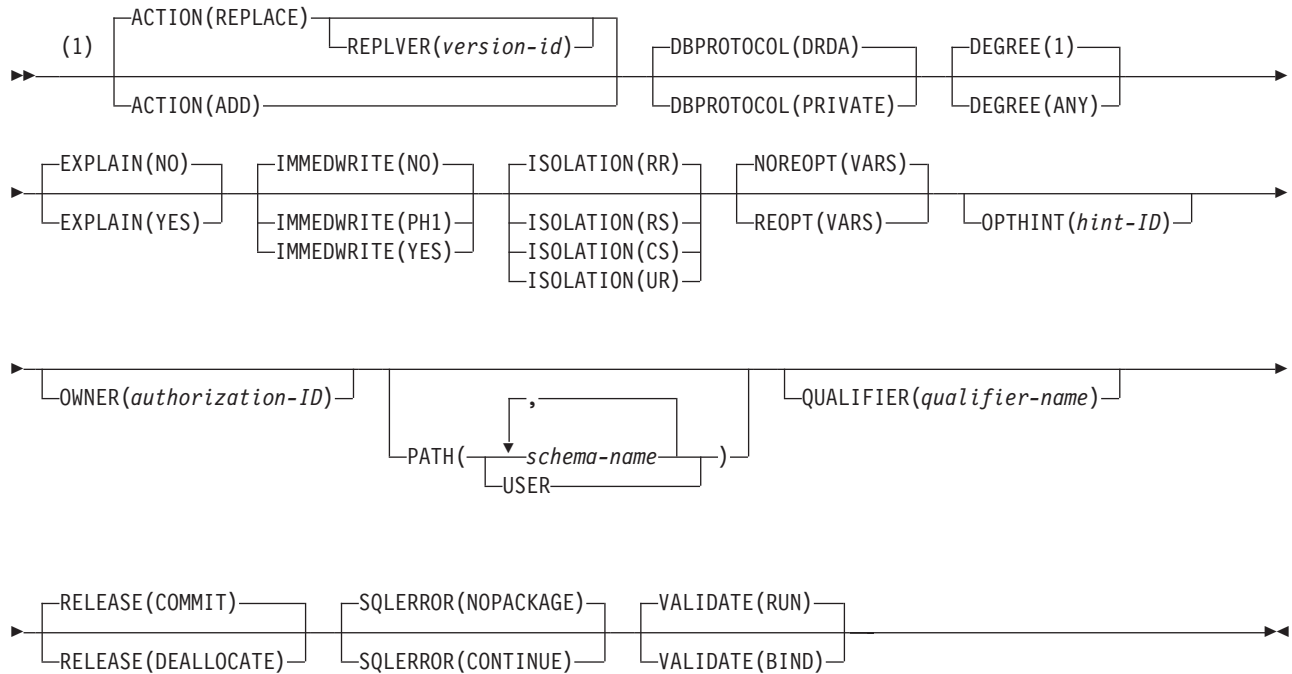
## db2sqljcustomize - SQLJ profile customizer



### *options-string:*



*DB2 for z/OS options:*

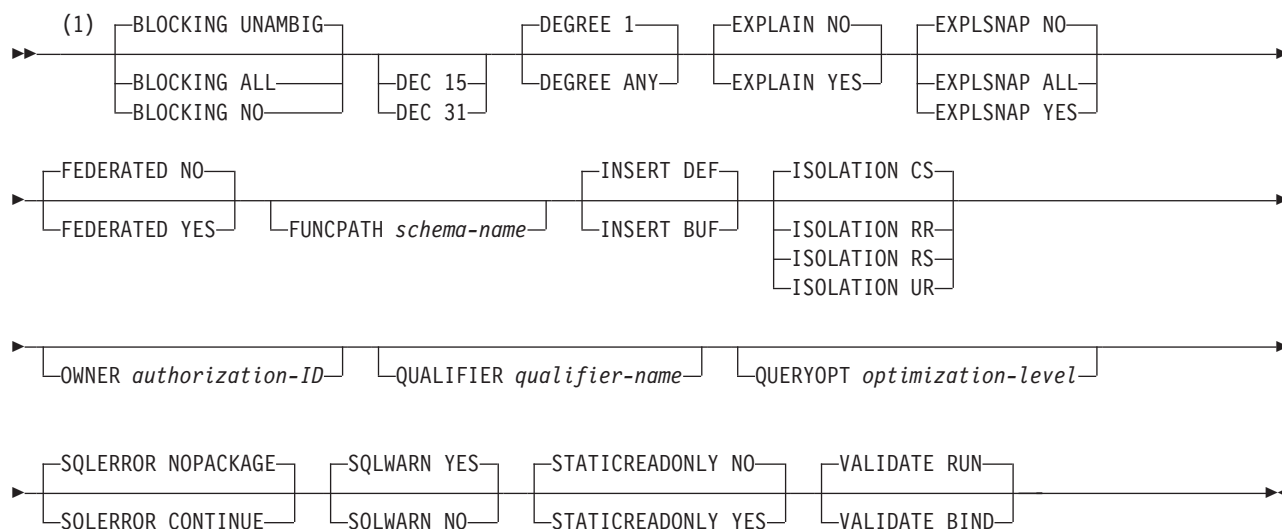


**Notes:**

- 1 These options can be specified in any order.

## db2sqljcustomize - SQLJ profile customizer

### DB2 Database for Linux, UNIX, and Windows options



#### Notes:

- 1 These options can be specified in any order.

#### Command parameters:

##### -help

Specifies that the SQLJ customizer describes each of the options that the customizer supports. If any other options are specified with -help, they are ignored.

##### -url

Specifies the URL for the data source for which the profile is to be customized. A connection is established to the data source that this URL represents if the -automaticbind or -onlinecheck option is specified as YES or defaults to YES. The variable parts of the -url value are:

##### server

The domain name or IP address of the MVS system on which the DB2 subsystem resides.

##### port

The TCP/IP server port number that is assigned to the DB2 subsystem. The default is 446.

##### database

A name for the database server for which the profile is to be customized.

If the connection is to a DB2 for z/OS server, *database* is the DB2 location name that is defined during installation. All characters in this value must be uppercase characters. You can determine the location name by executing the following SQL statement on the server:

```
SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1;
```

If the connection is to a DB2 Database for Linux, UNIX, and Windows server, *database* is the database name that is defined during installation.

If the connection is to an IBM Cloudscape server, the *database* is the fully-qualified name of the file that contains the database. This name must be enclosed in double quotation marks ("). For example:



"c:/databases/testdb"

*property=value;*

A property for the JDBC connection. For the definitions of these properties, see Properties for the IBM DB2 Driver for JDBC and SQLJ.

**-datasource** *JNDI-name*

Specifies the logical name of a DataSource object that was registered with JNDI. The DataSource object represents the data source for which the profile is to be customized. A connection is established to the data source if the `-automaticbind` or `-onlinecheck` option is specified as YES or defaults to YES. Specifying `-datasource` is an alternative to specifying `-url`. The DataSource object must represent a connection that uses IBM DB2 Driver for JDBC and SQLJ type 4 connectivity.

**-user** *user-ID*

Specifies the user ID to be used to connect to the data source for online checking or binding a package. You must specify `-user` if you specify `-url`. You must specify `-user` if you specify `-datasource`, and the DataSource object that *JNDI-name* represents does not contain a user ID.

**-password** *password*

Specifies the password to be used to connect to the data source for online checking or binding a package. You must specify `-password` if you specify `-url`. You must specify `-password` if you specify `-datasource`, and the DataSource object that *JNDI-name* represents does not contain a password.

**-automaticbind** YES|NO

Specifies whether the customizer binds DB2 packages at the data source that is specified by the `-url` parameter.

The default is YES.

The number of packages and the isolation levels of those packages are controlled by the `-rootpkgname` and `-singlepkgname` options.

Before the bind operation can work, the following conditions need to be met:

- TCP/IP and DRDA must be installed at the target data source.
- Valid `-url`, `-username`, and `-password` values must be specified.
- The `-username` value must have authorization to bind a package at the target data source.

**-pkgversion** AUTO|*version-id*

Specifies the package version that is to be used when packages are bound at the server for the serialized profile that is being customized. `db2sqljcustomize` stores the version ID in the serialized profile and in the DB2 package. Run-time version verification is based on the consistency token, not the version name. To automatically generate a version name that is based on the consistency token, specify `-pkgversion AUTO`.

The default is that there is no version.

**-bindoptions** *options-string*

Specifies a list of options, separated by spaces. These options have the same function as DB2 precompile and bind options with the same names. If you are preparing your program to run on a DB2 for z/OS system, specify DB2 for z/OS options. If you are preparing your program to run on a DB2 Database for Linux, UNIX, and Windows system, specify DB2 Database for Linux, UNIX, and Windows options.

*Notes on bind options:*

## db2sqljcustomize - SQLJ profile customizer

- Specify ISOLATION only if you also specify the -singlepkgname option.
- The value for STATICREADONLY is YES for servers that support STATICREADONLY, and NO for other servers. When you specify STATICREADONLY YES, DB2 processes ambiguous cursors as if they were read-only cursors. For troubleshooting iterator declaration errors, you need to explicitly specify STATICREADONLY NO, or declare iterators so that they are unambiguous. For example, if you want an iterator to be unambiguously updatable, declare the iterator to implement sqlj.runtime.ForUpdate. If you want an iterator to be read-only, include the FOR READ ONLY clause in SELECT statements that use the iterator.

**Important:** Specify only those program preparation options that are appropriate for the data source at which you are binding a package. Some values and defaults for the IBM DB2 Driver for JDBC and SQLJ are different from the values and defaults for DB2.

### **-storebindoptions**

Specifies that values for the -bindoptions and -staticpositioned parameters are stored in the serialized profile. If db2sqljbind is invoked without the -bindoptions or -staticpositioned parameter, the values that are stored in the serialized profile are used during the bind operation. When multiple serialized profiles are specified for one invocation of db2sqljcustomize, the parameter values are stored in each serialized profile. The stored values are displayed in the output from the db2sqljprint utility.

### **-collection** *collection-name*

The qualifier for the packages that db2sqljcustomize binds. db2sqljcustomize stores this value in the customized serialized profile, and it is used when the associated packages are bound. If you do not specify this parameter, db2sqljcustomize uses a collection ID of NULLID.

### **-onlinecheck** YES|NO

Specifies whether online checking of data types in the SQLJ program is to be performed. The -url or -datasource option determines the data source that is to be used for online checking. The default is YES if the -url or -datasource parameter is specified. Otherwise, the default is NO.

### **-qualifier** *qualifier-name*

Specifies the qualifier that is to be used for unqualified objects in the SQLJ program during online checking. This value is not used as the qualifier when the packages are bound.

### **-rootpkgname** | **-singlepkgname**

Specifies the names for the packages that are associated with the program. If -automaticbind is NO, these package names are used when db2sqljbind runs. The meanings of the parameters are:

#### **-rootpkgname** *package-name-stem*

Specifies that the customizer creates four packages, one for each of the four DB2 isolation levels. The names for the four packages are:

|                           |                        |
|---------------------------|------------------------|
| <i>package-name-stem1</i> | For isolation level UR |
| <i>package-name-stem2</i> | For isolation level CS |
| <i>package-name-stem3</i> | For isolation level RS |
| <i>package-name-stem4</i> | For isolation level RR |

If -longpkgname is not specified, *package-name-stem* must be an alphanumeric string of seven or fewer bytes.

If `-longpkgname` is specified, *package-name-stem* must be an alphanumeric string of 127 or fewer bytes.

**-singlepkgname** *package-name*

Specifies that the customizer creates one package, with the name *package-name*. If you specify this option, your program can run at only one isolation level. You specify the isolation level for the package by specifying the ISOLATION option in the `-bindoptions` options string.

If `-longpkgname` is not specified, *package-name* must be an alphanumeric string of eight or fewer bytes.

If `-longpkgname` is specified, *package-name* must be an alphanumeric string of 128 or fewer bytes.

Using the `-singlepkgname` option is not recommended.

If you do not specify `-rootpkgname` or `-singlepkgname`, `db2sqljcustomize` generates four package names that are based on the serialized profile name. A serialized profile name is of the following form:

*program-name\_SJProfileIDNumber.ser*

The four generated package names are of the following form:

*Bytes-from-program-nameIDNumberPkgIsolation*

Table 1 shows the parts of a generated package name and the number of bytes for each part.

The maximum length of a package name is *maxlen*. *maxlen* is 8 if `-longpkgname` is not specified. *maxlen* is 128 if `-longpkgname` is specified.

Table 1. Parts of a package name that is generated by `db2sqljcustomize`

| Package name part              | Number of bytes                                                                                          | Value                                                                                              |
|--------------------------------|----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|
| <i>Bytes-from-program-name</i> | $m = \min(\text{Length}(\textit{program-name}), \textit{maxlen} - 1 - \text{Length}(\textit{IDNumber}))$ | First <i>m</i> bytes of <i>program-name</i> , in uppercase                                         |
| <i>IDNumber</i>                | $\text{Length}(\textit{IDNumber})$                                                                       | <i>IDNumber</i>                                                                                    |
| <i>PkgIsolation</i>            | 1                                                                                                        | 1, 2, 3, or 4. This value represents the transaction isolation level for the package. See Table 2. |

Table 2 shows the values of the *PkgIsolation* portion of a package name that is generated by `db2sqljcustomize`.

Table 2. *PkgIsolation* values and associated isolation levels

| <i>PkgNumber</i> value | Isolation level for package |
|------------------------|-----------------------------|
| 1                      | Uncommitted read (UR)       |
| 2                      | Cursor stability (CS)       |
| 3                      | Read stability (RS)         |
| 4                      | Repeatable read (RR)        |

*Example:* Suppose that a profile name is `ThisIsMyProg_SJProfile111.ser`. The `db2sqljcustomize` option `-longpkgname` is not specified. Therefore, *Bytes-from-program-name* is the first four bytes of `ThisIsMyProg`, translated to uppercase, or `THIS`. *IDNumber* is 111. The four package names are:

## db2sqljcustomize - SQLJ profile customizer

```
THIS1111
THIS1112
THIS1113
THIS1114
```

*Example:* Suppose that a profile name is ThisIsMyProg\_SJProfile111.ser. The db2sqljcustomize option -longpkgname is specified. Therefore, *Bytes-from-program-name* is ThisIsMyProg, translated to uppercase, or THISISMYPROG. *IDNumber* is 111. The four package names are:

```
THISISMYPROG1111
THISISMYPROG1112
THISISMYPROG1113
THISISMYPROG1114
```

*Example:* Suppose that a profile name is A\_SJProfile0.ser. *Bytes-from-program-name* is A. *IDNumber* is 0. Therefore, the four package names are:

```
A01
A02
A03
A04
```

Letting db2sqljcustomize generate package names is not recommended. If any generated package names are the same as the names of existing packages, db2sqljcustomize overwrites the existing packages. To ensure uniqueness of package names, specify -rootpkgname.

### **-longpkgname**

Specifies that the names of the DB2 packages that db2sqljcustomize generates can be up to 128 bytes. Use this option only if you are binding packages at a server that supports long package names. If you specify -singlepkgname or -rootpkgname, you must also specify -longpkgname under the following conditions:

- The argument of -singlepkgname is longer than eight bytes.
- The argument of -rootpkgname is longer than seven bytes.

### **-staticpositioned NO|YES**

For iterators that are declared in the same source file as positioned UPDATE statements that use the iterators, specifies whether the positioned UPDATES are executed as statically bound statements. The default is NO. NO means that the positioned UPDATES are executed as dynamically prepared statements.

### **-tracefile** *file-name*

Enables tracing and identifies the output file for trace information. This option should be specified only under the direction of IBM Software Support.

### **-tracelevel**

If -tracefile is specified, indicates what to trace while db2sqljcustomize runs. The default is TRACE\_SQLJ. This option should be specified only under the direction of IBM Software Support.

### *serialized-profile-name* | *file-name*.grp

Specifies the names of one or more serialized profiles that are to be customized. A serialized profile name is of the following form:

```
program-name_SJProfileIDNumber.ser
```

You can specify the serialized profile name with or without the .ser extension.

*program-name* is the name of the SQLJ source program, without the extension .sqlj. *n* is an integer between 0 and *m-1*, where *m* is the number of serialized profiles that the SQLJ translator generated from the SQLJ source program.

## db2sqljcustomize - SQLJ profile customizer

You can specify serialized profile names in one of the following ways:

- List the names in the db2sqljcustomize command. Multiple serialized profile names must be separated by spaces.
- Specify the serialized profile names, one on each line, in a file with the name *file-name.grp*, and specify *file-name.grp* in the db2sqljcustomize command.

If you specify more than one serialized profile name, and if you specify or use the default value of -automaticbind YES, db2sqljcustomize binds a single DB2 package from the profiles. When you use db2sqljcustomize to create a single DB2 package from multiple serialized profiles, you must also specify the -rootpkgrname or -singlepkgrname option.

If you specify more than one serialized profile name, and you specify -automaticbind NO, if you want to bind the serialized profiles into a single DB2 package when you run db2sqljbind, you need to specify the same list of serialized profile names, in the same order, in db2sqljcustomize and db2sqljbind.

### Output:

When db2sqljcustomize runs, it creates a customized serialized profile. It also creates DB2 packages, if the automaticbind value is YES.

### Examples:

```
db2sqljcustomize -user richler -password mordecai
 -url jdbc:db2:/server:50000/sample -collection duddy
 -bindoptions "EXPLAIN YES" pgmname_SJProfile0.ser
```

### Usage notes:

**Online checking is always recommended:** It is highly recommended that you use online checking when you customize your serialized profiles. Online checking determines information about the data types and lengths of DB2 host variables, and is especially important for the following items:

- Predicates with java.lang.String host variables and CHAR columns  
Unlike character variables in other host languages, Java String host variables are not declared with a length attribute. To optimize a query properly that contains character host variables, DB2 needs the length of the host variables. For example, suppose that a query has a predicate in which a String host variable is compared to a CHAR column, and an index is defined on the CHAR column. If DB2 cannot determine the length of the host variable, it might do a table space scan instead of an index scan. Online checking avoids this problem by providing the lengths of the corresponding character columns.
- Predicates with java.lang.String host variables and GRAPHIC columns  
Without online checking, DB2 might issue a bind error (SQLCODE -134) when it encounters a predicate in which a String host variable is compared to a GRAPHIC column.
- Column names in the result table of an SQLJ SELECT statement at a remote server:  
Without online checking, the driver cannot determine the column names for the result table of a remote SELECT.

**Customizing multiple serialized profiles together:** Multiple serialized profiles can be customized together to create a single DB2 package. If you do this, and if you

## db2sqljcustomize - SQLJ profile customizer

specify `-staticpositioned YES`, any positioned UPDATE or DELETE statement that references a cursor that is declared *earlier in the package* executes statically, even if the UPDATE or DELETE statement is in a different source file from the cursor declaration. If you want `-staticpositioned YES` behavior when your program consists of multiple source files, you need to order the profiles in the `db2sqljcustomize` command to cause cursor declarations to be ahead of positioned UPDATE or DELETE statements in the package. To do that, list profiles that contain SELECT statements that assign result tables to iterators *before* profiles that contain the positioned UPDATE or DELETE statements that reference those iterators.

**Using a customized serialized profile at one data source that was customized at another data source:** You can run `db2sqljcustomize` to produce a customized serialized profile for an SQLJ program at one data source, and then use that profile at another data source. You do this by running `db2sqljbind` multiple times on customized serialized profiles that you created by running `db2sqljcustomize` once. When you run the programs at these data sources, the DB2 objects that the programs access must be identical at every data source. For example, tables at all data sources must have the same encoding schemes and the same columns with the same data types.

**Using the `-collection` parameter:** `db2sqljcustomize` stores the DB2 collection name in each customized serialized profile that it produces. When an SQLJ program is executed, the driver uses the collection name that is stored in the customized serialized profile to search for packages to execute. The name that is stored in the customized serialized profile is determined by the value of the `-collection` parameter. Only one collection ID can be stored in the serialized profile. However, you can bind the same serialized profile into multiple package collections by specifying the `COLLECTION` option in the `-bindoptions` parameter. To execute a package that is in a collection other than the collection that is specified in the serialized profile, include a `SET CURRENT PACKAGESET` statement in the program.

**Using the `VERSION` parameter:** Use the `VERSION` parameter to bind two or more versions of a package for the same SQLJ program into the same collection. You might do this if you have changed an SQLJ source program, and you want to run the old and new versions of the program.

To maintain two versions of a package, follow these steps:

1. Change the code in your source program.
2. Translate the source program to create a new serialized profile. Ensure that you do not overwrite your original serialized profile.
3. Run `db2sqljcustomize` to customize the serialized profile and create DB2 packages with the same package names and in the same collection as the original packages. Do this by using the same values for `-rootpkgname` and `-collection` when you bind the new packages that you used when you created the original packages. Specify the `VERSION` option in the `-bindoptions` parameter to put a version ID in the new customized serialized profile and in the new packages.

It is essential that you specify the `VERSION` option when you perform this step. If you do not, you overwrite your original packages.

When you run the old version of the program, DB2 loads the old versions of the packages. When you run the new version of the program, DB2 loads the new versions of the packages.

**Related reference:**

- “BIND” on page 355
- “db2sqljprint - SQLJ profile printer” on page 270
- “db2sqljbind - SQLJ profile binder” on page 252

### db2sqljprint - SQLJ profile printer

db2sqljprint prints the contents of a DB2 customized version of a profile as plain text.

**Authorization:**

None

**Command syntax:**

▶▶—db2sqljprint—*profilename*—————▶▶

**Command parameters:**

*profilename*

Specifies the relative or absolute name of an SQLJ profile file. When an SQLJ file is translated into a Java source file, information about the SQL operations it contains is stored in SQLJ-generated resource files called profiles. Profiles are identified by the suffix \_SJProfileN (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:**

```
db2sqljprint pgmname_SJProfile0.ser
```

**Related reference:**

- “db2sqljcustomize - SQLJ profile customizer” on page 259
- “db2sqljbind - SQLJ profile binder” on page 252



---

## db2start - Start DB2

Starts the current database manager instance background processes on a single database partition or on all the database partitions defined in a partitioned database 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.

The **db2start** command launches the DB2 product installation as a Windows service. The DB2 product installation on Windows can still be run as a process by specifying the /D switch when invoking **db2start**. The DB2 product installation can also be started as a service using the Control Panel or the **NET START** command.

Since **db2start** launches a Windows service, you must meet Windows requirements for starting a service. If Extended Security is disabled, you must be a member of the Administrators, Server Operators or Power Users group. If Extended Security is enabled, you must be a member of either the Administrators group or the DB2ADMNS group to start the database.

If a **db2start** operation in a multi-partition database is not completed within the value specified by the `start_stop_timeout` database manager configuration parameter, the database partitions that have timed out will be killed internally (all resources associated with the database partition will be removed). Environments with many database partitions with a low value for `start_stop_timeout` might experience this behavior. To resolve this behavior, increase the value of `start_stop_timeout`.

### Related reference:

- “START DATABASE MANAGER ” on page 728

### db2stop - Stop DB2

Stops the current database manager instance. **db2stop** can be executed as a system command or a CLP command.

If a **db2stop** operation in a multi-partition database is not completed within the value specified by the `start_stop_timeout` database manager configuration parameter, the database partitions that have timed out will be killed internally (all resources associated with the database partition will be removed). Environments with many database partitions with a low value for `start_stop_timeout` might experience this behavior. To resolve this behavior, increase the value of `start_stop_timeout`.

**Related reference:**

- “STOP DATABASE MANAGER ” on page 736

## db2support - Problem analysis and environment collection tool

Collects environment data about either a client or server machine and places the files containing system data into a compressed file archive.

This tool can also collect basic data about the nature of a problem through an interactive question and answer process with the user.

### Authorization:

For the most complete output, this utility should be invoked by the instance owner. Users with more limited privileges on the system can run this tool, however some of the data collection actions will result in reduced reporting and reduced output.

### Required connection:

None

### Command syntax:

```

▶▶ db2support output path
 [-a] [-r] [-cd current degree]
 [-cl collect level] [-co] [-cs current schema]
 [-d database name] [-f]
 [-c] [-u userid] [-p password]
 [-fp function path] [-g] [-h] [-il isolation level] [-l]
 [-m] [-n] [-ol optimization level] [-op optimization profile]
 [-ot optimization tables] [-q] [-ra refresh age] [-ro] [-s]
 [-se embedded SQL file] [-sf SQL file] [-st SQL statement]
 [-td termination character delimiter] [-v] [-x]

```

### Notes:

1. There is no separate option for invoking this tool in optimizer mode.
2. The db2support tool collects bad query-related information only if `-st`, `-sf`, or `-se` options are specified. In case there is an error or trap during optimization, `-cl 0` (collect level zero) should be used to collect all catalog tables and db2look table definitions without trying to explain a bad query. One of the four options mentioned here must be specified to work with optimizer problems.

## db2support - Problem Analysis and Environment Collection Tool

3. In case special registers have been set to values other than the default during the statement execution, it is very important for correct problem analysis that the same values should be passed as parameters to the db2support tool.

### Command parameters:

#### *output path*

Specifies the path where the archived library is to be created. This is the directory where user created files must be placed for inclusion in the archive.

#### **-a or -all\_core**

Specifies that all the core files are to be captured.

#### **-r or -recent\_core**

Specifies that the most recent core files are to be captured. This option is ignored if the -a option is specified.

#### **-c or -connect**

Specifies that an attempt be made to connect to the specified database.

#### **-cd or -curdegree**

Specifies the value of the current degree special register to use. The default is the value of the *dft\_degree* database configuration parameter.

#### **-cl or -collect**

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

- 0 = collect only catalogs, db2look, dbcfg, dbmcfg, db2set
- 1 = collect 0 plus exfmt
- 2 = collect 1 plus .db2service (this is the default)
- 3 = collect 2 plus db2batch

**-co** Collect catalogs for all tables in the database. The default is to only collect catalog information for the tables used in a query that has a problem.

#### **-cs or -curschema**

Specifies the value of the current schema to use to qualify any unqualified table names in the statement. The default value is the authorization ID of the current session user.

#### **-d database\_name or -database database\_name**

Specifies the name of the database for which data is being collected.

#### **-f or -flow**

Ignores pauses when requests are made for the user to Press <Enter> key to continue. This option is useful when running or calling the **db2support** tool via a script or some other automated procedure where unattended execution is desired.

#### **-fp or -funcpath**

Specifies value of the function path special register to be used to resolve unqualified user defined functions and types. The default value is "SYSIBM", "SYSFUN", "SYSPROC", X, where X is the value of the USER special register, delimited by double quotation marks.

#### **-g or -get\_dump**

Specifies that all files in a dump directory, excluding core files, are to be captured.

#### **-h or -help**

Displays help information. When this option is specified, all other options are ignored, and only the help information is displayed.

## db2support - Problem Analysis and Environment Collection Tool

### **-il or -isolation**

Specifies the isolation level to use to determine how data is locked and isolated from other processes while the data is being accessed. By default, the CURRENT ISOLATION special register is set to blanks.

### **-l or -logs**

Specifies that active logs are to be captured.

### **-m or -html**

Specifies that all system output is dumped into HTML formatted files. By default, all system related information is dumped into flat text files if this parameter is not used.

### **-n or -number**

Specifies the problem management report (PMR) number or identifier for the current problem.

### **-ol or -optlevel**

Specifies the value of the optimization level special register to use. The default is the value of the *dft\_queryopt* database configuration parameter.

### **-op or -optprofile**

Specifies value of the optimization profile special register to use. It is needed only if there was an optimization profile in effect when the statement was bound. The default is "" (an empty string).

### **-ot or -opttables**

Specifies the value of the special register called "CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION" that is used to identify the types of tables that can be considered when optimizing the processing of dynamic SQL queries. The initial value of CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION is "SYSTEM".

### **-p *password* or -password *password***

Specifies the password for the user ID.

### **-q or -question\_response**

Specifies that interactive problem analysis mode is to be used.

### **-ra or -refreshage**

Specifies the value of the refresh age special register. It applies only if there are materialized query tables (MQTs) that reference tables in the statement. The default value of CURRENT REFRESH AGE is zero.

### **-ro or -reopt**

Specifies whether EXPLAIN with REOPT ONCE should be used when explaining the query. The default is to ignore the REOPT ONCE option.

### **-s or -system\_detail**

Specifies that detailed hardware and operating system information is to be gathered.

### **-se *embedded SQL file* or -sqlembd *embedded SQL file***

Specifies the path of the embedded SQL file containing the SQL statement for which data is being collected.

### **-sf *SQL file* or -sqlfile *SQL file***

Specifies the file path containing the SQL statement for which data is being collected.

### **-st *SQL statement* or -sqlstmt *SQL statement***

Specifies the SQL statement for which data is being collected.

## db2support - Problem Analysis and Environment Collection Tool

### **-td or -delimiter**

Specifies the statement termination character. This command parameter works in the same way as the `-td` option of the `db2` command. The default is a semicolon.

### **-u userid or -user userid**

Specifies the user ID to connect to the database.

### **-v or -verbose**

Specifies that verbose output is to be used while this tool is running.

### **-x or -xml\_generate**

Specifies that an XML document containing the entire decision tree logic used during the interactive problem analysis mode (`-q` mode) is to be generated.

### **Examples:**

The `db2support` tool is invoked in the optimizer mode in one of the following ways:

- As an SQL statement from a command line.

```
db2support <output_directory> -d <database name> -st <sql_statement>
```

The `db2support` tool stores the query in the optimizer directory by copying the query into the file called "bad\_query.sql".

- As an SQL statement stored in a file.

```
db2support <output_directory> -d <database name> -sf <sql_file>
```

The file containing the query is copied by the tool into the optimizer directory.

- As a file containing an embedded static SQL statement with the query having the problem.

```
db2support <output_directory> -d <database name> -se <embedded_sql_file>
```

The file containing the query is copied by the tool into the optimizer directory. The file does not need to be in the current directory but should be readable by an invoking userID.

- While returning different levels of performance information.

```
db2support <output_directory> -d <database name> -collect 0
```

The `db2support` tool collects different levels of performance information based on the level of detail requested. The values 0 to 3 collect increasing amounts of detail. Catalog information and table definitions to enable you to reproduce the database objects for a production database are collected when a level of 0 is used.

To collect information to diagnose a slow query using optimizer-related special registers that were set by default, use:

```
db2support . -d sample -st "SELECT * FROM EMPLOYEE"
```

This example returns all the data to the `db2support.zip` file. Diagnostic files are created in the current directory and its subdirectories (since `.` is specified as the output path). The system information and diagnostic files are collected as well.

To collect the same information shown in the previous example but with the user-specified values for the optimizer-related special registers, use:

```
db2support . -d sample -st "SELECT * FROM EMPLOYEE" -cs db2usr -cd 3
-ol 5 -ra ANY -fp MYSCHEMA -op MYPROFSHEMA.MYPROFILE -ot ALL -il CS
```

## db2support - Problem Analysis and Environment Collection Tool

This example sets the following special registers: current schema to db2usr, current degree to 3, optimization level to 5, refresh age to ANY, function path to schema MYSCHEMA, optimization profile to MYPROFSCHEMA.MYPROFILE, current maintained table types to ALL, and the isolation level to CS. These values are set only for the connection that db2support establishes to the specified database and does not affect your entire environment. Providing the same special registry variables as used when the query was run is very important when correcting diagnostics.

### Usage notes:

In order to protect the security of business data, this tool does not collect table data, schema (DDL), or logs. Some of the options do allow for the inclusion of some aspects of schema and data (such as archived logs). Options that expose database schema or data should be used carefully. When this tool is invoked, a message is displayed that indicates how sensitive data is dealt with.

Data collected from the db2support tool will be from the machine where the tool runs. In a client-server environment, database-related information will be from the machine where the database resides via an instance attachment or connection to the database. For example, operating system or hardware information (-s option) and files from the diagnostic directory (DIAGPATH) will be from the local machine where the db2support tool is running. Data such as buffer pool information, database configuration, and table space information will be from the machine where the database physically resides.

There are some limitations on the type of queries accepted by the db2support optimizer tool:

- Multiple queries are not supported. If you place several queries in a file, the tool gathers all the objects necessary for each of the queries. However, only the last query is explained. This is also true for files containing embedded static SQL statements.
- The tool does not run customer applications. However, you can run the application at the same time you are running db2support provided you are using one of the three methods discussed to evaluate a particular bad or slow query.
- Stored procedures are not supported.

**db2support** does not collect explain data for dynamic SQL.

### Related concepts:

- “Combining DB2 database and OS diagnostics” in *Troubleshooting Guide*

### Related tasks:

- “Collecting environment information using db2support” in *Troubleshooting Guide*

### db2swtch - Switch default DB2 copy

Switches the default DB2 copy. The default DB2 copy is the copy that is used by applications that are not targeted at a specific DB2 copy. Issuing db2swtch launches the DB2 Copy Selection Wizard which you can follow to set a new default DB2 Copy. This command is only available on Windows operating systems.

**Authorization:**

*sysadm.*

**Required connection:**

None.

**Command syntax:**

```
►► db2swtch [-l] [-d installation-name]
```

**Command parameters:**

- l Displays a list of DB2 database product installations on the system.
- d *installation-name*  
Sets the default DB2 copy.

**Related reference:**

- “db2iupdt - Update instances ” on page 135



## db2sync - Start DB2 synchronizer

Facilitates the initial configuration of a satellite as well as changes to the configuration. This command can also be used to start, stop and monitor the progress of a synchronization session and to upload a satellite's configuration information (for example, communications parameters) to its control server.

### Authorization:

None

### Required connection:

None

### Command syntax:

```

▶▶—db2sync —————▶▶
 |
 | --t
 | --s application_version
 | --g

```

### Command parameters:

- t      Displays a graphical user interface that allows an administrator to change either the application version or synchronization credentials for a satellite.
- s **application\_version**  
      Sets the application version on the satellite.
- g      Displays the application version currently set on the satellite.

### Related reference:

- “db2SyncSatellite API - Start satellite synchronization” in *Administrative API Reference*
- “db2SyncSatelliteStop API - Pause satellite synchronization” in *Administrative API Reference*

### db2sysray - Start DB2 system tray

Starts the DB2 system tray tool. It is a Windows operating system notify icon which monitors the status of a DB2 database service on Windows operating systems. **db2sysray** provides a visual indication of when the service is started and stopped, as well as the ability to start and stop the service. It also provides a launch point for the DB2 Control Center.

The **db2sysray** icon has two modes, started and stopped. When the monitored instance is stopped, the icon contains an overlay with a red square. When the instance is started, the red square disappears.

In partitioned database environments, the **db2sysray** icon will be in started mode only when all partitions are started. If one or more partitions are stopped, the **db2sysray** icon will be in stopped mode.

When multiple DB2 copies are installed on a single Windows operating system, **db2sysray** can monitor DB2 instances for each DB2 copy that is installed. To monitor a non-default DB2 copy, you can execute the **db2sysray.exe** application from the SQLLIB/bin of the DB2 copy you wish to monitor.

You can monitor a single DB2 instance or multiple instances at the same time. Multiple instances can be monitored using multiple **db2sysray** processes. A separate icon will appear in the system tray for each instance monitored by **db2sysray**. Hovering over each icon with your mouse will display the name of the DB2 copy that is being monitored followed by the DB2 instance name monitored by that **db2sysray** icon.

The **db2sysray** icon can be launched manually from the DB2 command window by issuing the **db2sysray** command, or automatically when the Windows operating system starts. **db2sysray** is configured to start automatically when you install the DB2 database. However, having **db2sysray** configured to start automatically when the system starts, does not mean that it will attempt to start the DB2 service as well. All it means is that it will start monitoring the status of the DB2 database automatically.

Issuing the **db2idrop** command against an instance monitored by a running **db2sysray** process will force the **db2sysray** application to clean up its registry entries and exit.

**db2sysray** is only available on Windows platforms.

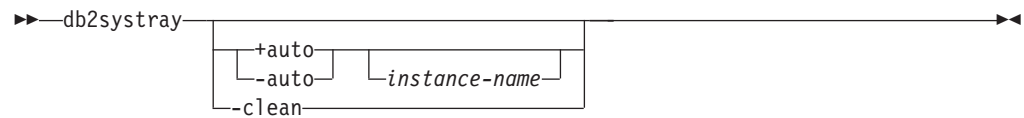
#### Authorization:

No special authority is required for starting **db2sysray**. Appropriate authority is required for taking actions.

#### Required connection:

None.

#### Command syntax:

**Command parameters:**

**+auto** Start **db2sysray** automatically for the specified instance when the Windows operating system starts. **db2sysray** can also be configured to launch automatically by enabling the Launch Tool at Startup **db2sysray** menu option.

**-auto** Disable **db2sysray** from starting automatically for the specified instance when the Windows operating system starts.

*instance-name*

Name of the DB2 instance to be monitored. If no instance name is specified, **db2sysray** will monitor the default local DB2 instance. If no instance exists, or the specified instance is not found, **db2sysray** will exit quietly.

**-clean** Clean up all registry entries for all DB2 instances monitored by **db2sysray** and stop all running **db2sysray.exe** processes.

**Examples:**

1. C:\SQLLIB\bin> db2sysray

Starts **db2sysray** for the default DB2 instance specified by the DB2INSTANCE environment variable.

2. C:\SQLLIB\bin\> db2sysray DB2INST1

Starts **db2sysray** for the instance named DB2INST1.

3. C:\SQLLIB\bin\> db2sysray +auto

Starts **db2sysray** for the default DB2 instance, and configures **db2sysray** to start monitoring this instance automatically when the Windows operating system starts.

4. C:\SQLLIB\bin\> db2sysray +auto DB2INST1

Starts **db2sysray** for the instance named DB2INST1, and configures **db2sysray** to start monitoring this instance automatically when the Windows operating system starts.

5. C:\SQLLIB\bin\> db2sysray -auto

Disables the auto start option for the default instance defined by the DB2INSTANCE environment variable.

6. C:\SQLLIB\bin\> db2sysray -auto DB2INST1

Disables the auto start option for instance DB2INST1.

7. C:\SQLLIB\bin\> db2sysray -clean

Removes all registry entries created by **db2sysray** and stops all running **db2sysray.exe** processes. If **db2sysray.exe** processes are running for other installed DB2 copies, they will not be cleaned up. You must execute **db2sysray -clean** from the SQLLIB/bin for each DB2 copy you wish to clean up.

**Related concepts:**

- “Control Center overview” in *Administration Guide: Implementation*

## db2tapemgr - Manage log files on tape

Allows the storage and retrieval of DB2 log files to and from tape. The location on tape is stored in the history file.

### Scope:

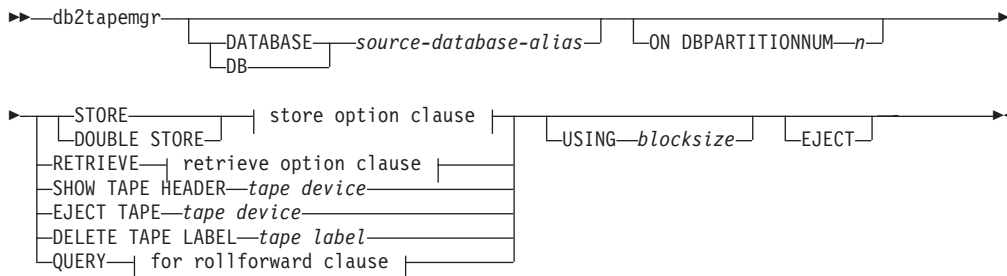
### Authorization:

One of the following:

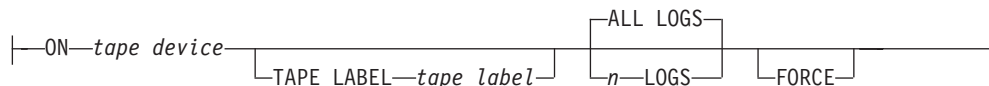
- *sysadm*
- *sysctrl*
- *sysmaint*

### Required connection:

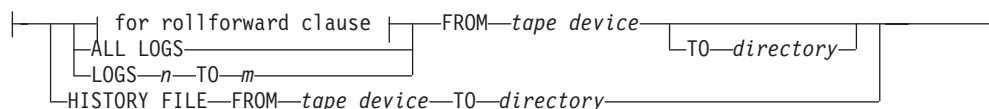
### Command syntax:



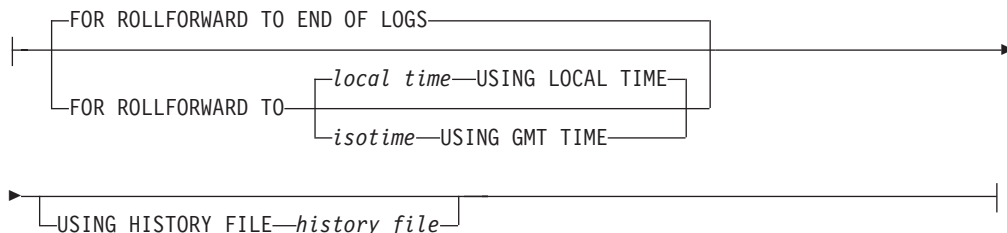
### store option clause:



### retrieve option clause:



### for rollforward clause:



### Command parameters:

**DATABASE** *source-database-alias*

Specifies the name of the database. If no value is specified, DB2DBDFT will be used. If no value is specified, and DB2DBDFT is not set, the operation fails.

**ON DBPARTITIONNUM**

Specifies the database partition number to work on. If no value is specified, DB2NODE is used.

**STORE ON** *tape device*

Stores log file to tape and deletes it.

**DOUBLE STORE ON** *tape device*

Stores all log files that have been stored only once and those log files never stored. Deletes only the log files that have been stored twice to tape; others are kept on disk.

**TAPE LABEL**

Specifies a label to be applied to the tape. If *tape label* is not specified, one will be generated automatically in the following format:  
*database-alias | timestamp* (up to 22 characters, up to 8 characters for the database alias and 14 characters for the time stamp in seconds).

**ALL LOGS or n LOGS**

Specifies that the command applies to all logs or a specified number of logs.

**FORCE**

Specifies that if the tape has not expired, then over write it.

**USING** *blocksize*

Specifies the block size for tape access. The default size is 5120, and it must be a multiple of 512. The minimum is 512.

**EJECT** Specifies that the tape is to be ejected after the operation completes.

**RETRIEVE FOR ROLLFORWARD TO**

Specifies that the utility will interactively prompt for all logs that are required for the specified rollforward and retrieve them from tape. If a directory is not specified, the path specified by the *overflowlogpath* configuration parameter is used. If a directory is not specified and *overflowlogpath* is not set, the operation fails.

**END OF LOGS**

Specifies that log files up to the end of the log will be retrieved.

*isotime* **USING GMT TIME**

Specifies that log files up to the time specified will be retrieved.

*local time* **USING LOCAL TIME**

Specifies that log files up to the time specified will be retrieved.

**USING HISTORY FILE** *history file*

Specifies an alternate history file to be used.

**FROM** *tape device*

Specifies the tape device to retrieve log files from.

**TO** *directory*

Specifies a directory to copy retrieved log files to.

**RETRIEVE ALL LOGS or LOGS n TO m**

Specifies that the command applies to all logs or a specified number of logs on a tape.

## db2tapemgr - Manage Log Files on Tape

**FROM** *tape device*

Specifies the tape device to retrieve log files from.

**TO** *directory*

Specifies a directory to copy retrieved log files to.

**RETRIEVE HISTORY FILE**

Retrieves the history file

**FROM** *tape device*

Specifies the tape device to retrieve log files from.

**TO** *directory*

Specifies a directory to copy retrieved log files to.

**SHOW TAPE HEADER** *tape device*

Shows the content of the tape header file DB2TAPEMGR.HEADER

**EJECT TAPE** *tape device*

Ejects the tape.

**DELETE TAPE LABEL** *tape label*

Deletes all locations from the history file that refer to the specified tape label.

**QUERY FOR ROLLFORWARD TO**

Displays the location of the log files that are required for rollforward.

**END OF LOGS**

*isotime* **USING GMT TIME**

Specifies that the operation should query the logs up to the time specified.

*local time* **USING LOCAL TIME**

Specifies that the operation should query the logs up to the time specified.

**USING HISTORY FILE** *history file*

Specifies an alternate history file to be used.

**Examples:**

**Related concepts:**

- “Log archiving using db2tapemgr” in *Data Recovery and High Availability Guide and Reference*

---

## db2tbst - Get table space state

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

**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

**Related reference:**

- “LIST TABLESPACES ” on page 550

### db2trc - Trace

Controls the trace facility of a DB2 instance or the DB2 Administration Server. The trace facility records information about operations and formats this information into readable form. Enabling the trace facility might impact your system's performance. As a result, only use the trace facility when directed by a DB2 Support technical support representative.

#### **Authorization:**

To trace a DB2 instance on a UNIX operating systems, one of the following:

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

To trace the DB2 Administration Server on a UNIX operating systems:

- *dasadm*

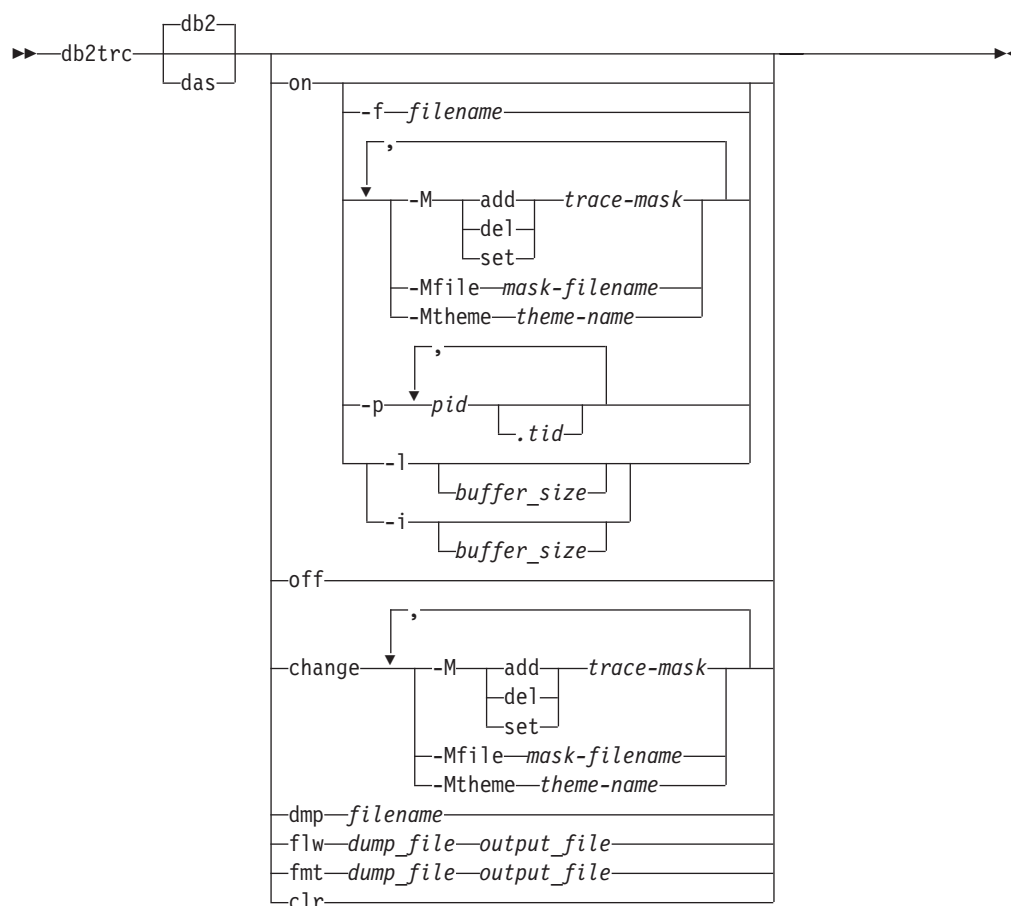
On a Windows operating system, no authorization is required.

#### **Required connection:**

None

#### **Command syntax:**



**Command parameters:**

- db2** Specifies that all trace operations will be performed on the DB2 instance. This is the default.
- das** Specifies that all trace operations will be performed on the DB2 Administration Server.
- on** Use this parameter to start the trace facility.

**-f filename**

Specifies that trace information should be continuously written to the specified file, until **db2trc** is turned off. Using this option can generate an extremely large dump file. Use this option only when instructed by DB2 Support.

**-M<action> trace-mask**

Enables a trace mask, which limits the operations recorded by the trace facility. Trace masks are provided by DB2 Support technical support representatives as necessary. The possible values for <action> are:

- add** Adds a trace mask element.
- del** Deletes a trace mask element.
- set** Sets the trace mask to a specific value. All previous contents of the mask are lost.

**-Mfile mask-filename**

Loads a list of trace mask actions from the file *mask-filename*.

**-Mtheme** *theme-name*

Loads a trace mask theme from a theme file. By default, the theme file in `~/sql11ib/cfg` is used.

**-p** *pid.tid*

Only enables the trace facility for the specified process IDs (*pid*) and thread IDs (*tid*). The period (`.`) must be included if a *tid* is specified. A maximum of five *pid.tid* combinations is supported.

For example, to enable tracing for processes 10, 20, and 30 the syntax is:

```
db2trc on -p 10,20,30
```

To enable tracing only for thread 33 of process 100 and thread 66 of process 200 the syntax is:

```
db2trc on -p 100.33,200.66
```

**-l** [*buffer\_size*] | **-i** [*buffer\_size*]

This option specifies the size and behavior of the trace buffer. '`l`' 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). The buffer size can be specified in either bytes or megabytes. To specify the buffer size in megabytes, add the character "`m`" to the buffer size. For example, to start **db2trc** with a 4-megabyte buffer:

```
db2trc on -l 4m
```

The default and maximum trace buffer sizes vary by platform. The minimum buffer size is 1 MB. The buffer size must be a power of 2.

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

```
db2trc off
```

**change**

Changes values associated with a trace mask on a trace that is already running. Trace masks are provided by DB2 Support technical support representatives as necessary.

**-Madd** *trace-mask*

Adds a trace mask element.

**-Mdel** *trace-mask*

Deletes a trace mask element.

**-Mset** *trace-mask*

Sets the trace mask to a specific value. All previous contents of the mask are lost.

**-Mfile** *mask-filename*

Loads a list of trace mask actions from the file *mask-filename*.

**-Mtheme** *theme-name*

Loads a trace mask theme from a theme file.

**dmp** Dumps the trace information to a file. The following command will put the information in the current directory in a file called `db2trc.dmp`:

```
db2trc dmp db2trc.dmp
```

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

**flw | fmt**

After the trace is dumped to a binary file, confirm that it is taken by formatting it into a text file. Use either the flw option (to format records sorted by process or thread), or the fmt option (to format records 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
```

**clr** Clears the contents of the trace buffer. This option can be used to reduce the amount of collected information. This option has no effect when tracing to a file.

**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.

The default and maximum trace buffer sizes vary by platform. The minimum buffer size is 1 MB.

When tracing the database server, it is recommended that the trace facility be turned on prior to starting the database manager.

**Related concepts:**

- “Dumping a DB2 trace file” in *Troubleshooting Guide*
- “Formatting a DB2 trace file” in *Troubleshooting Guide*
- “Obtaining a DB2 trace using db2trc” in *Troubleshooting Guide*

**Related reference:**

- “db2drdat - DRDA trace ” on page 86
- “SkipTrace CLI/ODBC configuration keyword” in *Call Level Interface Guide and Reference, Volume 1*

---

### db2uidl - 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.

#### Authorization:

*sysadm*

#### Required connection:

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

#### Command syntax:

```
db2uidl -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:

It is not necessary to use this tool unless there are indexes in the database that were created on a database running on a version of DB2 earlier than Version 5. This tool was not designed to handle certain types of names. If a specific table name or table schema is a delimited identifier containing lowercase characters, special characters, or blanks, it is preferable to request processing of *all* tables or schemas. The resulting output can be edited.

#### Related reference:

- "Conversion of type-1 indexes in migrated databases" in *Migration Guide*

## db2undgp - Revoke execute privilege

Revoke the execute privilege on external stored procedures. This command can be used against external stored procedures.

During the database migration, EXECUTE for all existing functions, methods, and External stored procedure is granted to PUBLIC. This will cause a security exposure for External Stored procedures that contain SQL data access. To prevent users from accessing SQL objects which the user might not have privilege for, use the db2undgp command.

### Command syntax:

```

▶▶ db2undgp [-d dbname] [-h] [-o outfile] [-r]

```

### Command parameters:

**-d** *dbname*

Specifies a database name whose maximum length is 8 characters.

**-h** Displays help for the command.

**-o** *outfile*

Output the revoke statements in the specified file. Length of the File name should be <= 80.

**-r** Perform the revoke

### Usage notes:

At least one of the -r or -o options must be specified.

### Related reference:

- “Changes to the EXECUTE privilege on PUBLIC for migrated routines” in *Migration Guide*

---

### db2unins - Uninstall DB2 database product

Uninstalls one or more DB2 database products. **db2unins** can be found both in the installation media and in a DB2 install copy on the system. If ran from the installation media, only the `-f`, `-l`, `-t` and `-?` parameters can be used. If ran from a DB2 install copy, all the options can be used.

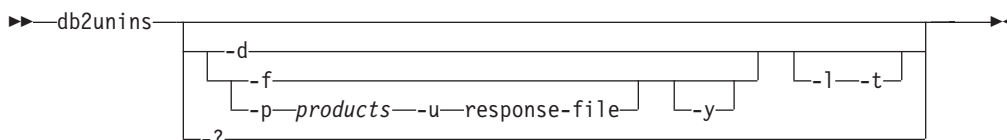
#### Authorization:

*sysadm.*

#### Required connection:

None.

#### Command syntax:



#### Command parameters:

Running the **db2unins** command without any of the `-?`, `-d`, `-p` or `-u` parameters will result in the removal of all DB2 database products under the current installation directory.

- d** Displays the products that are installed in the current DB2 copy on the system. This option is only available when executed from an installed copy of a DB2 database product.
- f** Performs a brute force uninstallation of all DB2 database products on the system. The **db2unins -f** command can be issued from either the installation media or an install copy on your machine. Your system will reboot when you successfully issue **db2unins -f**. It can only be issued if there are no DB2 products prior to version 9 installed on the system.
- p products** Specifies the products that should be uninstalled where *products* is a semicolon separated list of the abbreviations for DB2 database products enclosed in double quotes. For example, `-p "ESE";"PE";"QP"`. This option is only available when executed from an installed copy of a DB2 database product.
- u response-file** Performs an uninstallation based on what is specified in *response-file*. This option is also used to perform a silent uninstallation and is only available when executed from an installed copy of a DB2 database product.
- y** Ensures that no confirmation is done during the uninstallation process.
- l** Specifies the location of the log file.
- t** Turns on the trace functionality. The trace file will be used for debugging problems with the **db2unins** command.
- ?** Displays help for the **db2unins**

#### Related tasks:

## db2unins - Uninstall DB2 database product

- “Uninstalling a DB2 product using a response file (Windows)” in *Installation and Configuration Supplement*
- “Uninstalling your DB2 product (Windows)” in *Quick Beginnings for DB2 Servers*

### 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.

#### Related reference:

- "CREATE DATABASE " on page 395



## db2xprt - Format trap file

Formats the DB2 database binary trap files into a human readable ASCII file. Trap files (\*.TRP) are located in the instance directory (DB2INSTPROF) by default or in the diagnostic data directory path if the DIAGPATH database manager configuration parameter is set. It can be found under the SQLLIB/BIN directory. The **db2xprt** command uses DB2 symbol files (.PDB) in order to format the trap files.

### Authorization:

You must have access to the DIAGPATH directory.

### Command syntax:

```

▶▶—db2xprt—┬───────────────────────────────────infile──────────────────────────────────▶
 │
 │ ┌──/p──path──┐ ┌──/n──┐
 │ └──┬──┘ └──┬──┘
 │ ┌──/v──┐
 │ └──┬──┘
 │
 └──┬───outfile──┘

```

### Command parameters:

*/p path* A semicolon (;) separated path that points to the location or locations where the binary files and PDB files are located.

*/v* Displays version information.

*/n* Formats data without regard to line number information.

*infile* Specifies the input file.

*outfile* Specifies the output file.

### Examples:

If a trap file called DB30882416.TRP had been produced in your DIAGPATH, you could format it as follows:

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

### Related concepts:

- “Trap files” in *Troubleshooting Guide*

### Related tasks:

- “Formatting trap files (Windows)” in *Troubleshooting Guide*

---

### doce\_deinstall - Uninstall DB2 Information Center

Uninstalls the DB2 Information Center that is in the same install path as the **doce\_deinstall** tool. This command is only available on the Linux operating systems.

The **doce\_deinstall** command is located at `DB2DIR/doc/install`, where `DB2DIR` is the location where the current version of the DB2 Information Center is installed.

#### Authorization:

Root

#### Required Connection:

None.

#### Command syntax:

```
doce_deinstall [-a] [-l log-file] [-t trace-file] [-h|-?]
```

#### Command parameters:

- a** Removes the Information Center from its current location.
- l log-file**  
Specifies the log file. The default log file is `/tmp/doce_deinstall.log$$`, where `$$` is the process ID.
- t trace-file**  
Turns on the debug mode. The debug information is written to the file name specified as *trace-file*.
- h/-?** Displays usage information.

#### Examples:

- To uninstall DB2 Information Center that is installed in `/opt/ibm/db2/doce`, issue:

```
cd /opt/ibm/db2/doce
doce_deinstall -a
```

#### Related concepts:

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

#### Related tasks:

- “Removing DB2 products using the `db2_deinstall` or `doce_deinstall` command (Linux and UNIX)” in *Quick Beginnings for DB2 Servers*
- “Uninstalling your DB2 product (Linux and UNIX)” in *Quick Beginnings for DB2 Servers*

#### Related reference:

- “`db2_deinstall` - Uninstall DB2 products or features ” on page 10
- “`doce_install` - Install DB2 Information Center ” on page 297

## doce\_install - Install DB2 Information Center

Installs the DB2 Information Center. If no path is specified, the DB2 Information Center is installed by default in `/opt/ibm/db2ic/V9`. This command applies only to the Linux operating systems.

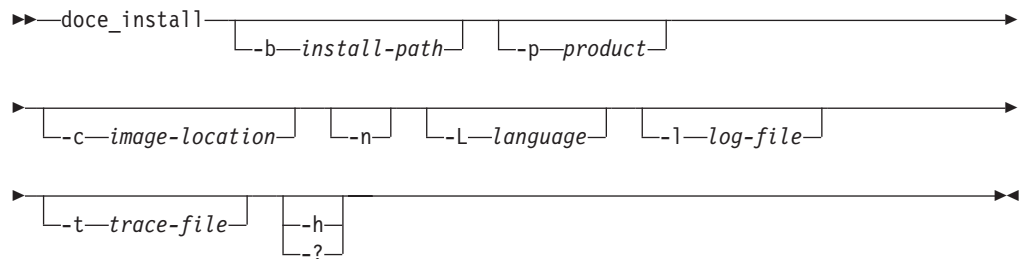
### Authorization:

Root

### Required Connection:

None.

### Command syntax:



### Command parameters:

#### **-b** *install-path*

Specifies the path where the DB2 Information Center is to be installed. *install-path* must be a full path name and its maximum length is limited to 128 characters. The default installation path is `/opt/ibm/db2ic/V9`. This parameter is mandatory when the `-n` parameter is specified.

#### **-p** *productID*

Specifies the *productID* of the DB2 Information Center. *productID* does not require DB2 as a prefix. This parameter is mandatory when the `-n` parameter is specified.

#### **-c** *image-location*

Specifies the product image location. To indicate multiple image locations, specify this parameter multiple times. For example, `-c CD1 -c CD2`. This parameter is only mandatory if the `-n` parameter is specified, your install requires more than one CD, and your images are not set up for automatic discovery. Otherwise, you are prompted for the location of the next CD at the time it is needed. For details on automatic discovery associated with multiple installation images, see Multiple CD installation (Linux and UNIX).

#### **-n** Specifies non-interactive mode.

#### **-L** *language*

Specifies national language support. The default is English. To install multiple languages at the same time, this parameter can be specified multiple times. For example, to install both English and German, specify `-L EN -L DE`.

## doce\_install - Install DB2 Information Center

- l** *log-file*  
Specifies the log file. The default log file is `/tmp/doce_install.log$$`, where `$$` is the process ID.
- t** *trace-file*  
Turns on the debug mode. The debug information is written to the file name specified as *trace-file*.
- h/-?** Displays usage information.

### Examples:

- To install from an image in `/mnt/cdrom`, and to be prompted for all needed input, issue:

```
cd /mnt/cdrom
./doce_install
```
- To install DB2 Information Center to `/db2/v9.1`, from an image in `/mnt/cdrom`, non-interactively in English, issue:

```
cd /mnt/cdrom
./doce_install -p doce -b /db2/v9.1 -n
```

### Related concepts:

- “Multiple CD installation (Linux and UNIX)” in *Quick Beginnings for DB2 Servers*

### Related tasks:

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

### Related reference:

- “db2\_install - Install DB2 product ” on page 11
- “doce\_deinstall - Uninstall DB2 Information Center ” on page 296

## disable\_MQFunctions

### Purpose

Disables the use of DB2 WebSphere MQ functions for the specified database.

### Authorization

One of the following:

- *sysadm*
- *dbadm*
- IMPLICIT\_SCHEMA on the database, if the implicit or explicit schema name of the function does not exist
- CREATEIN privilege on the schema, if the schema name, DB2MQ or DB2MQ1C exists

### Format

```

▶▶ disable_MQFunctions --n database --u userid --p password

```

A diagram shows a horizontal line with arrows at both ends. A bracket on the left side of the line points to the `--v` option. From this bracket, a vertical line descends to a box containing three options: `all`, `0pc`, and `1pc`.

### Parameters

#### **--n database**

Specifies the name of the database.

#### **--u userid**

Specifies the user ID used to connect to the database.

#### **--p password**

Specifies the password for the user ID.

- v** Optional. This is used for transactional and non-transactional user-defined function support. The values can be either `all`, `0pc`, or `1pc`. When you specify `0pc`, the disablement deletes from schema `db2mq`. If you specify `1pc`, then the disablement deletes from schema `db2mq1c`. If you specify `all`, then the disablement deletes from both schemas (`db2mq` and `db2mq1c`). If you do not specify this option, the disablement defaults to the `all` option.

### Example

In the following example, DB2MQ and DB2MQ1C functions are disabled for the database SAMPLE.

```
disable_MQFunctions -n sample -u user1 -p password1
```

### Related concepts:

- “How to use WebSphere MQ functions within DB2” in *Application Development Guide for Federated Systems*

## disable\_MQFunctions

### Related reference:

- “db2mqdsn - MQ listener ” on page 165
- “enable\_MQFunctions” on page 301
- “MQPUBLISH scalar function” in *Administrative SQL Routines and Views*
- “MQREAD scalar function” in *Administrative SQL Routines and Views*
- “MQREADALL table function” in *Administrative SQL Routines and Views*
- “MQREADALLCLOB table function” in *Administrative SQL Routines and Views*
- “MQREADCLOB scalar function” in *Administrative SQL Routines and Views*
- “MQRECEIVE scalar function” in *Administrative SQL Routines and Views*
- “MQRECEIVEALL table function” in *Administrative SQL Routines and Views*
- “MQRECEIVEALLCLOB table function” in *Administrative SQL Routines and Views*
- “MQRECEIVECLOB scalar function” in *Administrative SQL Routines and Views*
- “MQSEND scalar function” in *Administrative SQL Routines and Views*
- “MQSUBSCRIBE scalar function” in *Administrative SQL Routines and Views*
- “MQUNSUBSCRIBE scalar function” in *Administrative SQL Routines and Views*

## enable\_MQFunctions

Enables DB2 WebSphere MQ functions for the specified database and validates that the DB2 WebSphere MQ functions can be executed properly. The command fails if WebSphere MQ and WebSphere MQ AMI have not been installed and configured.

### Authorization:

One of the following:

- *sysadm*
- *dbadm*
- IMPLICIT\_SCHEMA on the database, if the implicit or explicit schema name of the function does not exist
- CREATEIN privilege on the schema, if the schema name, DB2MQ or DB2MQ1C, exists

### Command syntax:

```

▶▶ enable_MQFunctions --n database --u userid --p password
▶
└─q queuemanager┐ └─force┐ └─novalidate┐ └─v┐
└──────────┘ └──────────┘ └──────────┘ └─all┘
└──────────┘ └──────────┘ └──────────┘ └─0pc┘
└──────────┘ └──────────┘ └──────────┘ └─1pc┘

```

### Command parameters:

- n** Specifies the name of the database that you want to enable.
- u** Specifies the user ID to connect to the database.
- p** Specifies the password for the user ID.
- q** Optional. The queue manager name that supports the transactional MQ user-defined functions. If you do not specify a name, it is the default queue manager, DB2MQ\_DEFAULT\_MQM. If you use this option, the function assumes the use of a **-novalidate** parameter.
- force** Optional. The use of this option allows the utility program to ignore the existing MQ UDFs. In other words, the program drops any existing functions, before recreating any MQ UDFs. Without this option, the command will not proceed after it finds that the MQ UDFs already exist.
- novalidate** Optional. This specifies that there will not be any validation of the DB2 MQSeries functions.
- v** Optional. This is used for transactional and non-transactional user-defined function support. The values can be either *all*, *0pc*, or *1pc*. When you specify *0pc*, the enablement creates schema *db2mq*. If you specify *1pc*, then the enablement creates schema *db2mq1c*. If you specify *all*, then the enablement creates all schemas under user-defined functions (*db2mq* and *db2mq1c*). If you do not specify this option, the enablement defaults to the *all* option.

### Examples:

## enable\_MQFunctions

The following example enables the transactional and non-transactional user-defined functions. The user connects to the database SAMPLE.

```
enable_MQFunctions -n sample -u user1 -p password1
```

In the next example, the user connects to the database SAMPLE. The example creates DB2MQ1C functions with schema DB2MQ1C.

```
enable_MQFunctions -n sample -u user1 -p password1 -v 1pc
```

### Usage notes:

The DB2 MQ user-defined functions run under the schemas DB2MQ or DB2MQ1C which are automatically created by this command. Before executing this command:

- Ensure that WebSphere MQ and WebSphere Application Messaging Interface (AMI) are installed, and that the version of WebSphere MQ is 5.1 or higher.
- Ensure that the environment variable \$AMT\_DATA\_PATH is defined.
- If you want to use transactional MQ UDFs, make sure that the database is configured for federated operations. Do this with the following command  

```
update dbm cfg using federated yes
```
- Change the directory to the cfg subdirectory of the DB2PATH

On UNIX:

- Use db2set to add AMT\_DATA\_PATH to the DB2ENVLIST.
- Ensure that the user account associated with UDF execution is a member of the mqm group.
- Ensure that the user who will be calling this command is a member of the mqm group.

**Note:** AIX 4.2 is not supported by MQSeries 5.2.

### Related concepts:

- “How to use WebSphere MQ functions within DB2” in *Application Development Guide for Federated Systems*

### Related reference:

- “MQREADALL table function” in *Administrative SQL Routines and Views*
- “db2mqdsn - MQ listener ” on page 165
- “disable\_MQFunctions” on page 299
- “MQPUBLISH scalar function” in *Administrative SQL Routines and Views*
- “MQREAD scalar function” in *Administrative SQL Routines and Views*
- “MQREADALLCLOB table function” in *Administrative SQL Routines and Views*
- “MQREADCLOB scalar function” in *Administrative SQL Routines and Views*
- “MQRECEIVE scalar function” in *Administrative SQL Routines and Views*
- “MQRECEIVEALL table function” in *Administrative SQL Routines and Views*
- “MQRECEIVEALLCLOB table function” in *Administrative SQL Routines and Views*
- “MQRECEIVECLOB scalar function” in *Administrative SQL Routines and Views*
- “MQSEND scalar function” in *Administrative SQL Routines and Views*
- “MQSUBSCRIBE scalar function” in *Administrative SQL Routines and Views*
- “MQUNSUBSCRIBE scalar function” in *Administrative SQL Routines and Views*



## installFixPack - Update installed DB2 products

Update the installed DB2 product(s) in a given location to the level same as the image. If there are multi-copy DB2 products installed, the **installFixPack** command updates one copy at a time according to the path specified. This command can be found at the top directory in the image.

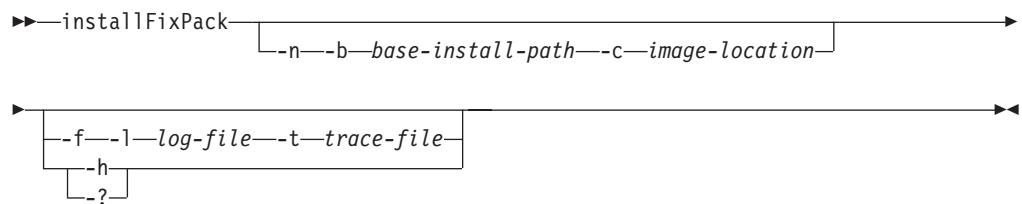
### Authorization:

Root authority.

### Required Connection:

None.

### Command syntax:



### Command parameters:

- n** Specifies non-interactive mode.
- b** *base-install-path*  
Specifies the path where the DB2 product that needs to be updated is installed. The length of the path is limited to 128 characters and is a full path name.
- c** *image-location*  
Specifies the image location. Mandatory when a single Fix Pack image spans more than one CD. To indicate multiple image locations, this parameter can be specified multiple times. For example, to indicate that the images are located on multiple CDs, specify `-c CD1 -c CD2`.
- f** Force option. This option forces a lower level fix pack image on top of a higher level fix pack image of the installed DB2 product or refreshes installed DB2 products to the same level. If the fix pack image is at a higher level than the installed DB2 product, this option is ignored.
- l** *log-file*  
Specifies the log file. The default log file is `/tmp/installFixPack.log$$`, where `$$` is the process id.
- t** *trace-file*  
Turns on the debug mode. The debug information is written to the file name specified.
- h/-?** Displays usage information.

### Examples:

- To perform an interactive update from GA to FP5 when DB2 ESE German is installed on `/opt/ibm/db2/V9.1`, from the FP5 image, issue:  

```
./installFixPack
```

## installFixPack - Update installed DB2 products

- To perform a silent update from GA to FP5 when DB2 ESE German is installed on /opt/ibm/db2/V9.1, from the FP5 image, issue:  

```
./installFixPack -b /opt/ibm/db2/V9.1 -c full_path_to_NLPACK_image -n
```
- If for any reason the installed DB2 product files get corrupted, instead of uninstalling and installing again to refresh the installation, issue:  

```
./installFixPack -f -b full_path_where_DB2_product_installed
```

### Related tasks:

- “Applying fix packs” in *Quick Beginnings for DB2 Servers*

## setup - Install DB2

Installs DB2 products. This command is only available on Windows operating systems. The command for UNIX operating systems is **db2setup**.

This utility is located on the DB2 installation media. It launches the DB2 Setup wizard to define the installation and install DB2 products. If invoked with the *-u* option, it performs an installation without further input, taking installation configuration information from a response file.

### Command syntax:

```

>> setup [-c] [-f] [-i language] [-l log-file] [-m]
 [-p install-directory] [-t trace-file] [-u response-file]
 [-n DB2-copy-name] [--?] [-h]

```

### Command parameters:

- c** Ensures that the setup.exe exits immediately after starting the installation. By selecting this option, the return code of the installation is not available when monitoring the exit code of setup.exe.
- f** Forces any DB2 processes to stop before installing.
- i language** Specifies the two-letter language code of the language in which to perform the installation.
- l log-file** Full path and file name of the log file to use.
- m** Used with *-u* option to show the progress dialog during the installation. However, it will not prompt for any input.
- p install-directory** Changes the installation path of the product. Specifying this option overrides the installation path that is specified in the response file.
- t trace-file** Generates a file with install trace information.
- u response-file** Specifies the full path and file name of the response file to use.
- n DB2-copy-name** Specifies the DB2 copy name that you want the install to use. Specifying this option overrides the installation path that is specified in the response file.
- , -h** Generates usage information.

### Related reference:

- “db2setup - Install DB2” on page 248

## setup - Install DB2

- “Language identifiers for running the DB2 Setup wizard in another language” in *Quick Beginnings for DB2 Servers*

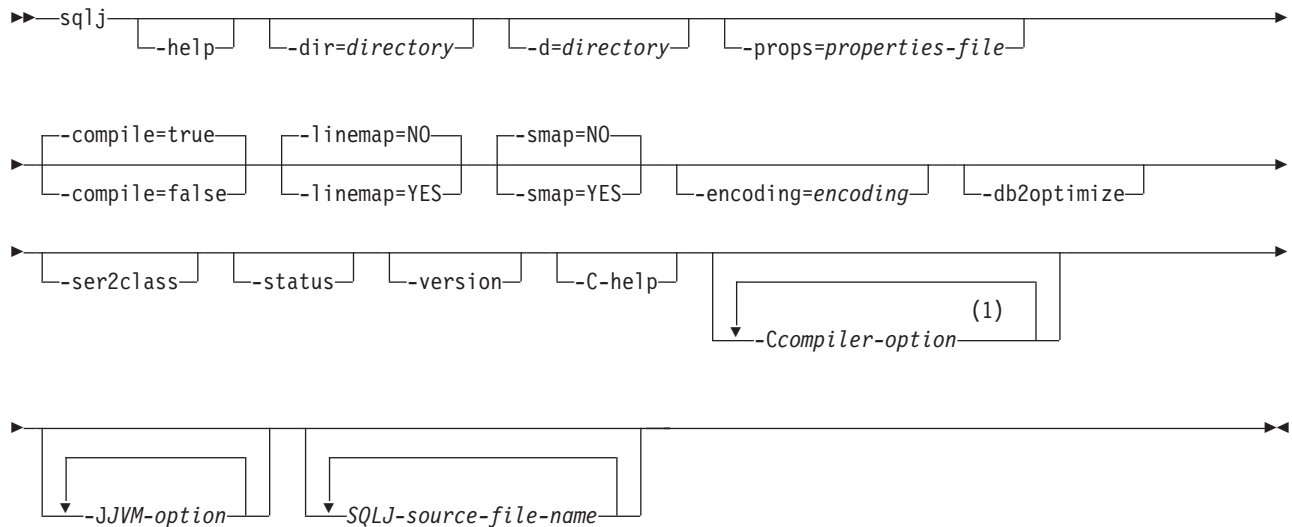
## sqlj - SQLJ translator

The sqlj command translates an SQLJ source file into a Java source file and zero or more SQLJ serialized profiles. By default, the sqlj command also compiles the Java source file.

### Authorization:

None

### Command syntax:



### Notes:

- 1 The `-C-classpath` and `-C-sourcepath` options are used by the SQLJ translator as well as by the Java compiler.

### Command parameters:

#### **-help**

Specifies that the SQLJ translator describes each of the options that the translator supports. If any other options are specified with `-help`, they are ignored.

#### **-dir=directory**

Specifies the name of the directory into which SQLJ puts `.java` files that are generated by the translator. The default directory is the directory that contains the SQLJ source files.

The translator uses the directory structure of the SQLJ source files when it puts the generated files in directories. For example, suppose that you want the translator to process two files:

- `file1.sqlj`, which is not in a Java package
- `file2.sqlj`, which is in Java package `sqlj.test`

Also suppose that you specify the parameter `-dir=/src` when you invoke the translator. The translator puts the Java source file for `file1.sqlj` in directory `/src` and puts the Java source file for `file2.sqlj` in directory `/src/sqlj/test`.

## sqlj - SQLJ translator

### **-d=directory**

Specifies the name of the directory into which SQLJ puts the binary files that are generated by the translator. These files include:

- The serialized profile files (.ser files)
- If the sqlj command invokes the Java compiler, the class files that are generated by the compiler (.class files)

The default directory is the directory that contains the SQLJ source files.

The translator uses the directory structure of the SQLJ source files when it puts the generated files in directories. For example, suppose that you want the translator to process two files:

- file1.sqlj, which is not in a Java package
- file2.sqlj, which is in Java package sqlj.test

Also suppose that you specify the parameter `-d=/src` when you invoke the translator. The translator puts the serialized profiles for file1.sqlj in directory `/src` and puts the serialized profiles for file2.sqlj in directory `/src/sqlj/test`.

### **-props=properties-file**

Specifies the name of a file from which the SQLJ translator is to obtain a list of options.

### **-compile=true | false**

Specifies whether the SQLJ translator compiles the generated Java source into bytecodes.

#### **true**

The translator compiles the generated Java source code. This is the default.

#### **false**

The translator does not compile the generated Java source code.

### **-linemap=no | yes**

Specifies whether line numbers in Java exceptions match line numbers in the SQLJ source file (the .sqlj file), or line numbers in the Java source file that is generated by the SQLJ translator (the .java file).

**no** Line numbers in Java exceptions match line numbers in the Java source file. This is the default.

#### **yes**

Line numbers in Java exceptions match line numbers in the SQLJ source file.

### **-smap=no | yes**

Specifies whether the SQLJ translator generates a source map (SMAP) file for each SQLJ source file. An SMAP file is used by some Java language debug tools. This file maps lines in the SQLJ source file to lines in the Java source file that is generated by the SQLJ translator. The file is in the Unicode UTF-8 encoding scheme. Its format is described by Original Java Specification Request (JSR) 45, which is available from this web site:

<http://www.jcp.org>

**no** Do not generated SMAP files. This is the default.

#### **yes**

Generate SMAP files. An SMAP file name is *SQLJ-source-file-name.java.smap*. The SQLJ translator places the SMAP file in the same directory as the generated Java source file.

**-encoding=*encoding-name***

Specifies the encoding of the source file. Examples are JIS or EUC. If this option is not specified, the default converter for the operating system is used.

**-db2optimize**

Specifies that the SQLJ translator generates code for a connection context class that is optimized for DB2. `-db2optimize` optimizes the code for the user-defined context but not the default context. When you run the SQLJ translator with the `-db2optimize` option, the IBM DB2 Driver for JDBC and SQLJ file `db2jcc.jar` must be in the CLASSPATH for compiling the generated Java application.

**-ser2class**

Specifies that the SQLJ translator converts `.ser` files to `.class` files.

**-status**

Specifies that the SQLJ translator displays status messages as it runs.

**-version**

Specifies that the SQLJ translator displays the version of the IBM DB2 Driver for JDBC and SQLJ. The information is in this form:

```
IBM SQLJ xxxx.xxxx.xx
```

**-C-help**

Specifies that the SQLJ translator displays help information for the Java compiler.

**-C*compiler-option***

Specifies a valid Java compiler option that begins with a dash (-). Do not include spaces between `-C` and the compiler option. If you need to specify multiple compiler options, precede each compiler option with `-C`. For example:

```
-C-g -C-verbose
```

All options are passed to the Java compiler and are not used by the SQLJ translator, **except** for the following options:

**-classpath**

Specifies the user class path that is to be used by the SQLJ translator and the Java compiler. This value overrides the CLASSPATH environment variable.

**-sourcepath**

Specifies the source code path that the SQLJ translator and the Java compiler search for class or interface definitions. The SQLJ translator searches for `.sqlj` and `.java` files only in directories, not in JAR or zip files.

**-J*JVM-option***

Specifies an option that is to be passed to the Java virtual machine (JVM) in which the `sqlj` command runs. The option must be a valid JVM option that begins with a dash (-). Do not include spaces between `-J` and the JVM option. If you need to specify multiple JVM options, precede each compiler option with `-J`. For example:

```
-J-Xmx128m -J-Xmine2M
```

***SQLJ-source-file-name***

Specifies a list of SQLJ source files to be translated. This is a required parameter. All SQLJ source file names must have the extension `.sqlj`.

**Output:**

## sqlj - SQLJ translator

For each source file, *program-name.sqlj*, the SQLJ translator produces the following files:

- The generated source program  
The generated source file is named *program-name.java*.
- A serialized profile file for each connection context class that is used in an SQLJ executable clause  
A serialized profile name is of the following form:  
*program-name\_SJProfileIDNumber.ser*
- If the SQLJ translator invokes the Java compiler, the class files that the compiler generates.

### Examples:

```
sqlj -encoding=UTF8 -C-0 MyApp.sqlj
```

### Related reference:

- “db2sqljbind - SQLJ profile binder” on page 252
- “db2sqljcustomize - SQLJ profile customizer” on page 259
- “db2sqljprint - SQLJ profile printer” on page 270



---

## Chapter 2. Command Line Processor (CLP)

---

### db2 - Command line processor invocation

The **db2** command starts the command line processor (CLP). 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:

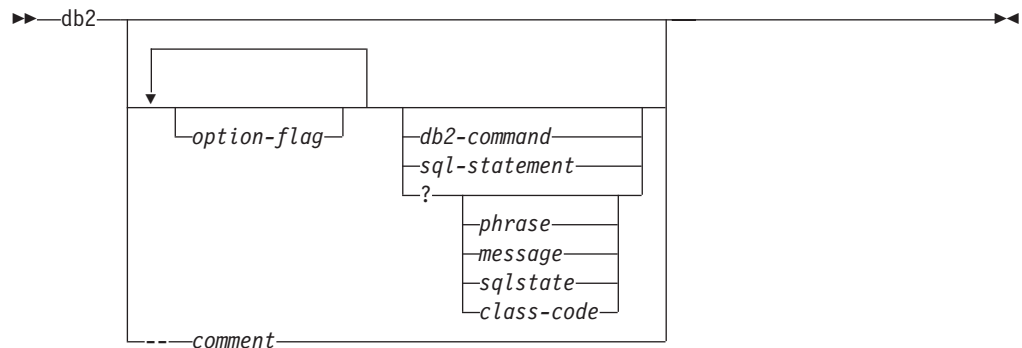
- Interactive input mode, characterized by the **db2 =>** input prompt
- Command mode, where each command must be prefixed by **db2**
- Batch mode, which uses the **-f** file input option.

On Windows operating systems, **db2cmd** 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 stops the command line processor. TERMINATE also stops the command line processor, but removes the associated back-end process and frees any memory that is being used. It is recommended that a TERMINATE be issued prior to every STOP DATABASE MANAGER (db2stop) command. It might also be necessary for a TERMINATE to be issued after database configuration parameters have been changed, in order for these changes to take effect. 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 Windows operating systems (!ls on UNIX, and !dir on Windows operating systems, for example).

#### Command Syntax:



#### **option-flag**

Specifies a CLP option flag.

#### **db2-command**

Specifies a DB2 command.

#### **sql-statement**

Specifies an SQL statement.

?

Requests CLP general help.

## db2 - Command Line Processor Invocation

### ? 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.

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

### Related reference:

- “Command line processor options” on page 312
- “Command line processor return codes” on page 320
- “Command line processor features” on page 321

---

## 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. Change an option setting from the interactive input mode or a command file using UPDATE COMMAND OPTIONS.

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 3 on page 313 summarizes the CLP option flags. These options can be specified in 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 3. 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              |
| -d            | This option tells the command line processor to retrieve and display XML declarations of XML data.                                                                                             | OFF             |
| -e{c s}       | This option tells the command line processor to display SQLCODE or SQLSTATE. These options are mutually exclusive.                                                                             | OFF             |
| -filename     | This option tells the command line processor to read command input from a file instead of from standard input.                                                                                 | OFF             |
| -i            | This option tells the command line processor to 'pretty print' the XML data with proper indentation. This option will only affect the result set of XQuery statements.                         | OFF             |
| -lfilename    | This option tells the command line processor to log commands in a history file.                                                                                                                | OFF             |
| -m            | This option tells the command line processor to print the number of rows affected for INSERT/DELETE/UPDATE/MERGE.                                                                              | OFF             |
| -n            | Removes the new line character within a single delimited token. If this option is not specified, the new line character is replaced with a space. This option must be used with the -t option. | OFF             |
| -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              |
| -q            | This option tells the command line processor to preserve whitespaces and linefeeds in strings delimited with single or double quotes. When option q is ON, option n is ignored.                | OFF             |
| -xfilename    | 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 or -tdxx | This option tells the command line processor to define and to use x or xx as the statement termination character or characters (1 or 2 characters in length).                                  | 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 FETCH/SELECT warning messages.                                                                                                         | ON              |

## Command line processor options

Table 3. CLP Command Options (continued)

| Option Flag | Description                                                                                                                                                                                                                                                   | Default Setting |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| -x          | This option tells the command line processor to return data without any headers, including column names. This flag will not affect all commands. It applies to SQL statements and some commands that are based on SQL statements such as <b>LIST TABLES</b> . | OFF             |
| -zfilename  | 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.

### XML Declaration Option (-d):

The -d option tells the command line processor whether to retrieve and display XML declarations of XML data.

If set ON (-d), the XML declarations will be retrieved and displayed. If set OFF (+d or -d-), the XML declarations will not be retrieved and displayed. The default setting for this command option is OFF.

The XML declaration option does not affect any other command line processor options.

### 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 can 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
```

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 the **QUIT** command or **TERMINATE** command 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.

## Command line processor options

Input file lines which begin with (= are treated as the beginning of a comment block. Lines which end with =) mark the end of a comment block. The block of input lines that begins at (= and ends at =) is treated as a continuous comment by the command line processor. Spaces before (= and after =) are allowed. Comments may be nested, and may be used nested in statements. The command termination character (;) cannot be used after =).

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

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

### Pretty Print Option (-i):

The *-i* option tells the command line processor to 'pretty print' the XML data with proper indentation. This option will only affect the result set of XQuery statements.

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

The pretty print option does not affect any other command line processor options.

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

The *-lfilename* 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 can 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 -tv1 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 Number of Rows Affected Option (-m):

The *-m* option tells the command line processor whether or not to print the number of rows affected for **INSERT**, **DELETE**, **UPDATE**, or **MERGE**.

If set ON (*-m*), the number of rows affected will be displayed for the statement of **INSERT/DELETE/UPDATE/MERGE**. If set OFF (+*m* or -*m*-), the number of rows affected will not be displayed. For other statements, this option will be ignored. The default setting for this command option is OFF.

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

### Remove New Line Character Option (-n):

Removes the new line character within a single delimited token. If this option is not specified, the new line character is replaced with a space. This option cannot be changed from within the interactive mode.

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

This option must be used with the `-t` option; see the option description for details.

### 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 might be affected by the `+o` option:

- `-rfilename`: 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 `-rfilename` 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 `-rfilename` option is specified.

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

### Preserve Whitespaces and Linefeeds Option (-q):

The `-q` option tells the command line processor to preserve whitespaces and linefeeds in strings delimited with single or double quotes.

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

If option `-q` is `ON`, option `-n` is ignored.

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

The `-rfilename` 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 can 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.

## Command line processor options

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 -rfilename 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 4. 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. This option cannot be changed from within the interactive mode.

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

To define termination characters 1 or 2 characters in length, use -td followed by the chosen character or characters. For example, -td%% sets %% as the statement termination characters. Alternatively, use the --#SET TERMINATOR directive to set the statement termination characters. For example, --#SET TERMINATOR%% sets %% as the statement termination characters.

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 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 instructs the command line processor on whether or not to display warning messages that may occur during a query (FETCH/SELECT). Warnings can occur during various stages of the query execution which may result in the messages being displayed before, during or after the data is returned. To ensure the data returned does not contain warning message text this flag can be used.

The default setting for this command option is ON.

### Suppress Printing of Column Headings Option (-x):

The -x option tells the command line processor to return data without any headers, including column names. This flag will not affect all commands. It applies to SQL statements and some commands that are based on SQL statements such as LIST TABLES.

The default setting for this command option is OFF.

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

The -zfilename 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 can 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 -zfilename 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.

### Related reference:

- “db2 - Command line processor invocation” on page 311
- “Command line processor return codes” on page 320

### Command line processor return codes

When the command line processor finishes processing a command or an SQL statement, it returns a return (or exit) 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.

#### Related reference:

- “db2 - Command line processor invocation” on page 311
- “Command line processor options” on page 312

## Command line processor features

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 can enter more commands.

Certain CLP commands and SQL statements require that the server instance is running and a database connection exists. Connect to a database by doing one of the following:

- Issue the SQL statement `DB2 CONNECT TO database`.
- Establish an implicit connection to the default database defined by the DB2 registry variable `DB2DBDFT`.

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 different 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`.

### Customizing the Command Line Processor:

It is possible to customize the interactive input prompt by using the `DB2_CLPPROMPT` registry variable. This registry variable can be set to any text string of maximum length 100 and can contain the tokens `%i`, `%ia`, `%d`, `%da` and `%n`. Specific values will be substituted for these tokens at run-time.

Table 5. `DB2_CLPPROMPT` tokens and run-time values

| DB2_CLPPROMPT token | Value at run-time                                                                                                                                                                                                                                                                                   |
|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>%ia</code>    | Authorization ID of the current instance attachment                                                                                                                                                                                                                                                 |
| <code>%i</code>     | Local alias of the currently attached instance. If no instance attachment exists, the value of the <code>DB2INSTANCE</code> registry variable. On Windows platforms only, if the <code>DB2INSTANCE</code> registry variable is not set, the value of the <code>DB2INSTDEF</code> registry variable. |
| <code>%da</code>    | Authorization ID of the current database connection                                                                                                                                                                                                                                                 |
| <code>%d</code>     | Local alias of the currently connected database. If no database connection exists, the value of the <code>DB2DBDFT</code> registry variable.                                                                                                                                                        |
| <code>%n</code>     | New line                                                                                                                                                                                                                                                                                            |

## Command line processor features

- If any token has no associated value at run-time, the empty string is substituted for that token.
- The interactive input prompt will always present the authorization IDs, database names, and instance names in upper case, so as to be consistent with the connection and attachment information displayed at the prompt.
- If the DB2\_CLPPROMPT registry variable is changed within CLP interactive mode, the new value of DB2\_CLPPROMPT will not take effect until CLP interactive mode has been closed and reopened.

### Examples:

If DB2\_CLPPROMPT is defined as (%ia%i, %da%d), the input prompt will have the following values:

- No instance attachment and no database connection. DB2INSTANCE set to "DB2". DB2DBDFT is not set.  
(@DB2, @)
- (Windows) No instance attachment and no database connection. DB2INSTANCE and DB2DBDFT not set. DB2INSTDEF set to "DB2".  
(@DB2, @)
- No instance attachment and no database connection. DB2INSTANCE set to "DB2". DB2DBDFT set to "SAMPLE".  
(@DB2, @SAMPLE)
- Instance attachment to instance "DB2" with authorization ID "tyronnem". DB2INSTANCE set to "DB2". DB2DBDFT set to "SAMPLE".  
(TYRONNEM@DB2, @SAMPLE)
- Database connection to database "sample" with authorization ID "horman". DB2INSTANCE set to "DB2". DB2DBDFT set to "SAMPLE".  
(@DB2, HORMAN@SAMPLE)
- Instance attachment to instance "DB2" with authorization ID "tyronnem". Database connection to database "sample" with authorization ID "horman". DB2INSTANCE set to "DB2". DB2DBDFT not set.  
(TYRONNEM@DB2, HORMAN@SAMPLE)

### 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**. 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 6. 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 reestablishes 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.

## Command line processor features

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 acknowledgment 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`.

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 upper case 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.

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 \'.

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 `stafflist.txt` in the `mydata` directory:

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

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

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

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

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.

To correctly display the national characters for single byte (SBCS) languages from the DB2 command line processor window, a True Type font must be selected. For example, in a Windows environment, open the command window properties notebook and select a font such as Lucinda Console.

### Related concepts:

- "DB2 Command Line Processor (CLP)" in *Developing SQL and External Routines*

## Command line processor features

### Related reference:

- “Command line processor options” on page 312
- “Command line processor return codes” on page 320

---

## Command Line Processor Help

### Invoking message help from the command line processor

Message help describes the cause of a message and describes any action you should take in response to the error.

#### Procedure:

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

```
? XXXnnnnn
```

where *XXXnnnnn* represents a valid message identifier.

For example, ? SQL30081 displays help about the SQL30081 message.

#### Related concepts:

- “Introduction to Messages” in *Message Reference Volume 1*

#### Related reference:

- “db2 - Command line processor invocation” on page 311

### Invoking command help from the command line processor

Command help explains the syntax of commands in the command line processor.

#### Procedure:

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

```
? command
```

where *command* represents a keyword or the entire command.

For example, ? catalog displays help for all of the CATALOG commands, while ? catalog database displays help only for the CATALOG DATABASE command.

#### Related tasks:

- “Invoking message help from the command line processor” on page 326
- “Accessing help from a DB2 tool, window, wizard or advisor” in *Online DB2 Information Center*
- “Displaying SQL state help from the command line processor” on page 833
- “Starting the DB2 Information Center” in *Online DB2 Information Center*

#### Related reference:

- “db2 - Command line processor invocation” on page 311



---

## Chapter 3. CLP Commands

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

Slashes (/) in directory paths are specific to UNIX-based systems, and are equivalent to back slashes (\) in directory paths on Windows operating systems.

---

### DB2 CLP Commands

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

*Table 7. DB2 CLP Commands*

| <b>CLP Session Control</b>                           |
|------------------------------------------------------|
| "LIST COMMAND OPTIONS " on page 522                  |
| "UPDATE COMMAND OPTIONS " on page 768                |
| "CHANGE ISOLATION LEVEL " on page 392                |
| "SET RUNTIME DEGREE " on page 719                    |
| "TERMINATE " on page 744                             |
| "QUIT " on page 618                                  |
| <b>Database Manager Control</b>                      |
| "START DATABASE MANAGER " on page 728                |
| "STOP DATABASE MANAGER " on page 736                 |
| "GET DATABASE MANAGER CONFIGURATION " on page 463    |
| "RESET DATABASE MANAGER CONFIGURATION " on page 669  |
| "UPDATE DATABASE MANAGER CONFIGURATION " on page 775 |
| "AUTOCONFIGURE " on page 346                         |
| <b>Database Control</b>                              |
| "RESTART DATABASE " on page 673                      |
| "CREATE DATABASE " on page 395                       |
| "DROP DATABASE " on page 426                         |
| "MIGRATE DATABASE " on page 579                      |
| "ACTIVATE DATABASE " on page 330                     |
| "DEACTIVATE DATABASE " on page 411                   |
| "QUIESCE " on page 612                               |
| "UNQUIESCE " on page 754                             |
| "LIST INDOUBT TRANSACTIONS " on page 538             |
| "LIST DRDA INDOUBT TRANSACTIONS " on page 533        |
| "GET DATABASE CONFIGURATION " on page 457            |
| "RESET DATABASE CONFIGURATION " on page 667          |
| "UPDATE DATABASE CONFIGURATION " on page 772         |
| "AUTOCONFIGURE " on page 346                         |
| <b>Database Directory Management</b>                 |

## DB2 CLP Commands

Table 7. DB2 CLP Commands (continued)

|                                                 |
|-------------------------------------------------|
| "CATALOG DATABASE " on page 372                 |
| "UNCATALOG DATABASE " on page 745               |
| "CATALOG DCS DATABASE " on page 375             |
| "UNCATALOG DCS DATABASE " on page 747           |
| "CHANGE DATABASE COMMENT " on page 390          |
| "LIST DATABASE DIRECTORY " on page 523          |
| "LIST DCS DIRECTORY " on page 531               |
| <b>ODBC Management</b>                          |
| "CATALOG ODBC DATA SOURCE " on page 386         |
| "LIST ODBC DATA SOURCES " on page 544           |
| "UNCATALOG ODBC DATA SOURCE " on page 753       |
| "GET CLI CONFIGURATION " on page 451            |
| "UPDATE CLI CONFIGURATION " on page 766         |
| <b>Client/Server Directory Management</b>       |
| "CATALOG LOCAL NODE " on page 382               |
| "CATALOG NAMED PIPE NODE " on page 384          |
| "CATALOG TCPIP/TCPIP4/TCPIP6 NODE " on page 387 |
| "UNCATALOG NODE " on page 752                   |
| "LIST NODE DIRECTORY " on page 541              |
| <b>Network Support</b>                          |
| "REGISTER " on page 637                         |
| "DEREGISTER " on page 415                       |
| "UPDATE LDAP NODE " on page 780                 |
| "CATALOG LDAP DATABASE " on page 377            |
| "UNCATALOG LDAP DATABASE " on page 749          |
| "CATALOG LDAP NODE " on page 381                |
| "UNCATALOG LDAP NODE " on page 751              |
| "REFRESH LDAP " on page 636                     |
| <b>DB2 Administration Server</b>                |
| "GET ADMIN CONFIGURATION " on page 441          |
| "RESET ADMIN CONFIGURATION " on page 663        |
| "UPDATE ADMIN CONFIGURATION " on page 756       |
| "CREATE TOOLS CATALOG " on page 408             |
| "DROP TOOLS CATALOG " on page 429               |
| <b>Recovery</b>                                 |
| "ARCHIVE LOG " on page 341                      |
| "BACKUP DATABASE " on page 349                  |
| "RECONCILE " on page 623                        |
| "RESTORE DATABASE " on page 675                 |
| "ROLLFORWARD DATABASE " on page 691             |
| "LIST HISTORY " on page 535                     |

Table 7. DB2 CLP Commands (continued)

|                                                        |
|--------------------------------------------------------|
| "PRUNE HISTORY/LOGFILE " on page 607                   |
| "UPDATE HISTORY " on page 778                          |
| "INITIALIZE TAPE " on page 511                         |
| "REWIND TAPE " on page 690                             |
| "SET TAPE POSITION " on page 723                       |
| <b>Operational Utilities</b>                           |
| "FORCE APPLICATION " on page 439                       |
| "LIST PACKAGES/TABLES " on page 545                    |
| "REORGCHK " on page 654                                |
| "REORG INDEXES/TABLE " on page 644                     |
| "RUNSTATS " on page 702                                |
| <b>Database Monitoring</b>                             |
| "GET MONITOR SWITCHES " on page 478                    |
| "UPDATE MONITOR SWITCHES " on page 782                 |
| "GET DATABASE MANAGER MONITOR SWITCHES " on page 468   |
| "GET SNAPSHOT " on page 487                            |
| "RESET MONITOR " on page 671                           |
| "INSPECT " on page 512                                 |
| "LIST ACTIVE DATABASES " on page 518                   |
| "LIST APPLICATIONS " on page 520                       |
| "LIST DCS APPLICATIONS " on page 529                   |
| <b>Data Utilities</b>                                  |
| "EXPORT " on page 433                                  |
| "IMPORT " on page 494                                  |
| "LOAD " on page 557                                    |
| "LOAD QUERY " on page 576                              |
| <b>Health Center</b>                                   |
| "ADD CONTACT " on page 333                             |
| "ADD CONTACTGROUP " on page 335                        |
| "DROP CONTACT " on page 424                            |
| "DROP CONTACTGROUP " on page 425                       |
| "GET ALERT CONFIGURATION" on page 443                  |
| "GET CONTACTGROUP " on page 454                        |
| "GET CONTACTGROUPS " on page 455                       |
| "GET CONTACTS " on page 456                            |
| "GET DESCRIPTION FOR HEALTH INDICATOR " on page 471    |
| "GET HEALTH NOTIFICATION CONTACT LIST " on page 473    |
| "GET HEALTH SNAPSHOT " on page 474                     |
| "GET RECOMMENDATIONS FOR HEALTH INDICATOR" on page 481 |
| "RESET ALERT CONFIGURATION " on page 665               |
| "UPDATE ALERT CONFIGURATION " on page 758              |

Table 7. DB2 CLP Commands (continued)

|                                                        |
|--------------------------------------------------------|
| "UPDATE CONTACT " on page 770                          |
| "UPDATE CONTACTGROUP " on page 771                     |
| "UPDATE HEALTH NOTIFICATION CONTACT LIST " on page 777 |
| <b>Application Preparation</b>                         |
| "PRECOMPILE " on page 583                              |
| "BIND" on page 355                                     |
| "REBIND " on page 619                                  |
| <b>Remote Server Utilities</b>                         |
| "ATTACH " on page 344                                  |
| "DETACH " on page 423                                  |
| <b>Table Space Management</b>                          |
| "LIST TABLESPACE CONTAINERS " on page 548              |
| "SET TABLESPACE CONTAINERS " on page 721               |
| "LIST TABLESPACES " on page 550                        |
| "QUIESCE TABLESPACES FOR TABLE " on page 615           |
| <b>Database Partition Management</b>                   |
| "ADD DBPARTITIONNUM " on page 336                      |
| "DROP DBPARTITIONNUM VERIFY " on page 428              |
| "LIST DBPARTITIONNUMS " on page 528                    |
| <b>Database Partition Group Management</b>             |
| "LIST DATABASE PARTITION GROUPS " on page 526          |
| "REDISTRIBUTE DATABASE PARTITION GROUP " on page 633   |
| <b>Data Links</b>                                      |
| <b>Additional Commands</b>                             |
| "DESCRIBE " on page 417                                |
| "ECHO " on page 431                                    |
| "GET AUTHORIZATIONS " on page 449                      |
| "GET CONNECTION STATE " on page 453                    |
| "GET INSTANCE " on page 477                            |
| "GET ROUTINE " on page 485                             |
| "HELP " on page 492                                    |
| "PING " on page 581                                    |
| "PUT ROUTINE " on page 609                             |
| "QUERY CLIENT " on page 611                            |
| "SET CLIENT " on page 716                              |

---

## 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:

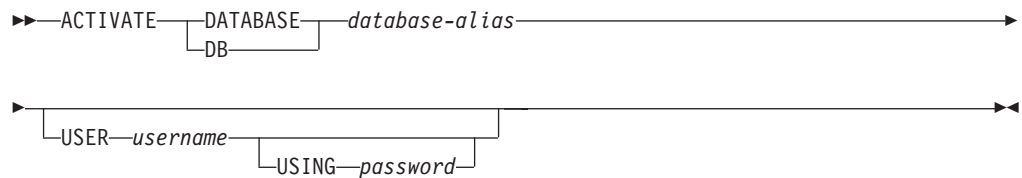
One of the following:

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

### Required connection:

None

### Command syntax:



### 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 using the `DEACTIVATE DATABASE` command, or using the `db2stop` command.

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

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.

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

**Related concepts:**

- “Quick-start tips for performance tuning” in *Performance Guide*

**Related reference:**

- “STOP DATABASE MANAGER ” on page 736
- “DEACTIVATE DATABASE ” on page 411
- “sqlc\_activate\_db API - Activate database” in *Administrative API Reference*

## ADD CONTACT

The command adds a contact to the contact list which can be either defined locally on the system or in a global list. Contacts are users to whom processes such as the Scheduler and Health Monitor send messages. The setting of the Database Administration Server (DAS) *contact\_host* configuration parameter determines whether the list is local or global.

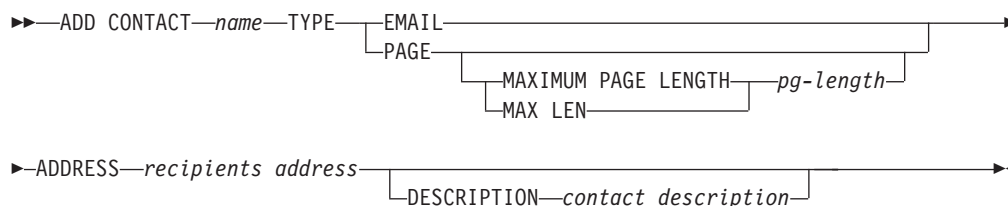
### Authorization:

None.

### Required connection:

None. Local execution only: this command cannot be used with a remote connection.

### Command syntax:



### Command parameters:

#### CONTACT name

The name of the contact that will be added. By default the contact will be added in the local system, unless the DB2 administration server configuration parameter *contact\_host* points to another system.

**TYPE** Method of contact, which must be one of the following two:

#### EMAIL

This contact wishes to be notified by e-mail at (ADDRESS).

**PAGE** This contact wishes to be notified by a page sent to ADDRESS.

#### MAXIMUM PAGE LENGTH *pg-length*

If the paging service has a message-length restriction, it is specified here in characters.

The notification system uses the SMTP protocol to send the notification to the mail server specified by the DB2 Administration Server configuration parameter *smtp\_server*. It is the responsibility of the SMTP server to send the e-mail or call the pager.

#### ADDRESS recipients-address

The SMTP mailbox address of the recipient. For example, *joe@somewhere.org*. The *smtp\_server* DAS configuration parameter must be set to the name of the SMTP server.

#### DESCRIPTION contact description

A textual description of the contact. This has a maximum length of 128 characters.

## ADD CONTACT

### Related tasks:

- “Enabling health alert notification” in *System Monitor Guide and Reference*

### Related reference:

- “db2AddContact API - Add a contact to whom notification messages can be sent” in *Administrative API Reference*
- “ADD CONTACT command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*



## ADD CONTACTGROUP

Adds a new contact group to the list of groups defined on the local system. A contact group is a list of users and groups to whom monitoring processes such as the Scheduler and Health Monitor can send messages. The setting of the Database Administration Server (DAS) *contact\_host* configuration parameter determines whether the list is local or global.

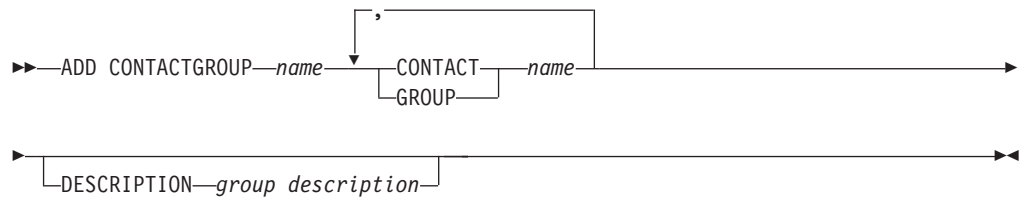
### Authorization:

None

### Required connection:

None. Local execution only: this command cannot be used with a remote connection.

### Command Syntax:



### Command Parameters:

#### **CONTACTGROUP** *name*

Name of the new contact group, which must be unique among the set of groups on the system.

#### **CONTACT** *name*

Name of the contact which is a member of the group. A contact can be defined with the `ADD CONTACT` command after it has been added to a group.

#### **GROUP** *name*

Name of the contact group of which this group is a member.

#### **DESCRIPTION** *group description*

Optional. A textual description of the contact group.

### Related tasks:

- “Enabling health alert notification” in *System Monitor Guide and Reference*

### Related reference:

- “db2AddContactGroup API - Add a contact group to whom notification messages can be sent” in *Administrative API Reference*
- “ADD CONTACTGROUP command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

## ADD DBPARTITIONNUM

Adds a new database partition server to the partitioned database environment. This command also creates a database partition for all databases on the new database partition server. The user can specify the source database partition server for the definitions of any system temporary table spaces to be created with the new database partition, or specify that no system temporary table spaces are to be created. The command must be issued from the database partition server that is being added.

### Scope:

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

### Authorization:

One of the following:

- *sysadm*
- *sysctrl*

### Required connection:

None

### Command syntax:

```

▶▶—ADD DBPARTITIONNUM—┬──LIKE DBPARTITIONNUM—db-partition-number—┘
 └──WITHOUT TABLESPACES—┘

```

### Command parameters:

#### LIKE DBPARTITIONNUM *db-partition-number*

Specifies that the containers for the new system temporary table spaces are the same as the containers of the database at the database partition server specified by *db-partition-number*. The database partition server specified must already be defined in the `db2nodes.cfg` file.

For system temporary table spaces that are defined to use automatic storage (in other words, system temporary table spaces that were created with the `MANAGED BY AUTOMATIC STORAGE` clause of the `CREATE TABLESPACE` statement or where no `MANAGED BY CLAUSE` was specified at all), the containers will not necessarily match those from the partition specified. Instead, containers will automatically be assigned by the database manager based on the storage paths that are associated with the database. This may or may not result in the same containers being used on these two partitions.

#### WITHOUT TABLESPACES

Specifies that containers for the system temporary table spaces are not created for any of the database partitions. The `ALTER TABLESPACE` statement must be used to add system temporary table space containers to each database partition before the database can be used.

If no option is specified, containers for the system temporary table spaces will be the same as the containers on the catalog partition for each database. The catalog partition can be a different database partition for

each database in the partitioned database environment. This option is ignored for system temporary table spaces that are defined to use automatic storage (in other words, system temporary table spaces that were created with the `MANAGED BY AUTOMATIC STORAGE` clause of the `CREATE TABLESPACE` statement or where no `MANAGED BY CLAUSE` was specified at all). For these table spaces, there is no way to defer container creation. Containers will automatically be assigned by the database manager based on the storage paths that are associated with the database.

**Usage notes:**

Before adding a new database partition server, ensure that there is sufficient storage for the containers that must be created for all databases in the instance.

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

If an add database partition server 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 database partition server being added. Existing database partitions remain unaffected on all other database partition servers. If the clean-up phase fails, no further clean up is done, and an error is returned.

The database partitions on the new database partition cannot contain user data until after the `ALTER DATABASE PARTITION GROUP` statement has been used to add the database partition to a database partition group.

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

This command will fail, if at any time in a database in the system a user table with an XML column has been created, successfully or not, or an XSR object has been registered, successfully or not.

To determine whether or not a database is enabled for automatic storage, **ADD DBPARTITIONNUM** has to communicate with the catalog partition for each of the databases in the instance. If automatic storage is enabled then the storage path definitions are retrieved as part of that communication. Likewise, if system temporary table spaces are to be created with the database partitions, **ADD DBPARTITIONNUM** might have to communicate with another database partition server to retrieve the table space definitions for the database partitions that reside on that server. The *start\_stop\_time* database manager configuration parameter is used to specify the time, in minutes, by which the other database partition server must respond with the automatic storage and table space definitions. If this time is exceeded, the command fails. If this situation occurs, increase the value of *start\_stop\_time*, and reissue the command.

**Compatibilities:**

For compatibility with versions earlier than Version 8:

- The keyword `NODE` can be substituted for `DBPARTITIONNUM`.

**Related concepts:**

## ADD DBPARTITIONNUM

- “Automatic storage databases” in *Administration Guide: Implementation*

### Related reference:

- “START DATABASE MANAGER ” on page 728

---

## ADD XMLSCHEMA DOCUMENT

Adds one or more XML schema documents to an existing but incomplete XML schema before completing registration.

### Authorization:

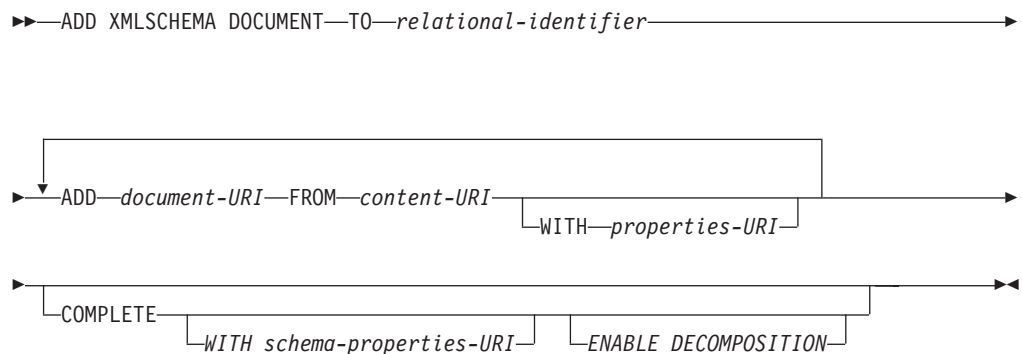
The following authority is required:

- The user ID must be the owner of the XSR object as recorded in the catalog view SYSCAT.XSROBJECTS.

### Required connection:

Database

### Command syntax:



### Description:

#### **TO** *relational-identifier*

Specifies the relational name of a registered but incomplete XML schema to which additional schema documents are added.

#### **ADD** *document-URI*

Specifies the uniform resource identifier (URI) of an XML schema document to be added to this schema, as the document would be referenced from another XML document.

#### **FROM** *content-URI*

Specifies the URI where the XML schema document is located. Only a file scheme URI is supported.

#### **WITH** *properties-URI*

Specifies the URI of a properties document for the XML schema. Only a file scheme URI is supported.

#### **COMPLETE**

Indicates that there are no more XML schema documents to be added. If specified, the schema is validated and marked as usable if no errors are found.

#### **WITH** *schema-properties-URI*

Specifies the URI of a properties document for the XML schema. Only a file scheme URI is supported.

## ADD XMLSCHEMA DOCUMENT

### ENABLE DECOMPOSITION

Specifies that this schema is to be used for decomposing XML documents.

### Example:

```
ADD XMLSCHEMA DOCUMENT TO JOHNDOE.PRODSHEMA
 ADD 'http://myPOschema/address.xsd'
 FROM 'file:///c:/TEMP/address.xsd'
```

### Related reference:

- “COMPLETE XMLSCHEMA ” on page 394
- “REGISTER XMLSCHEMA ” on page 640

## ARCHIVE LOG

Closes and truncates the active log file for a recoverable database.

### Authorization:

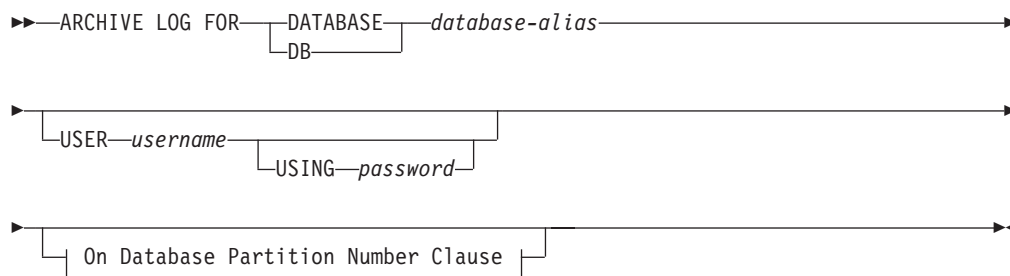
One of the following:

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

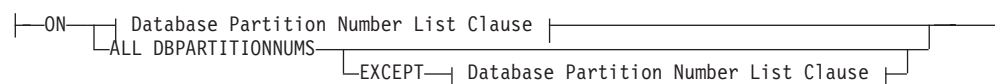
### Required connection:

None. This command establishes a database connection for the duration of the command.

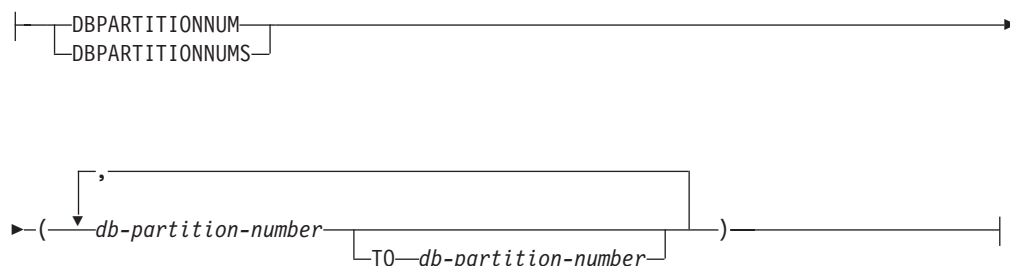
### Command syntax:



### On Database Partition Number Clause:



### Database Partition Number List Clause:



### Command parameters:

#### DATABASE database-alias

Specifies the alias of the database whose active log is to be archived.

#### USER username

Identifies the user name under which a connection will be attempted.

## ARCHIVE LOG

### **USING password**

Specifies the password to authenticate the user name.

### **ON ALL DBPARTITIONNUMS**

Specifies that the command should be issued on all database partitions in the `db2nodes.cfg` file. This is the default if a database partition number clause is not specified.

### **EXCEPT**

Specifies that the command should be issued on all database partitions in the `db2nodes.cfg` file, except those specified in the database partition number list.

### **ON DBPARTITIONNUM/ON DBPARTITIONNUMS**

Specifies that the logs should be archived for the specified database on a set of database partitions.

### **db-partition-number**

Specifies a database partition number in the database partition number list.

### **TO db-partition-number**

Used when specifying a range of database partitions for which the logs should be archived. All database partitions from the first database partition number specified up to and including the second database partition number specified are included in the database partition number list.

### **Usage notes:**

This command can be used to collect a complete set of log files up to a known point. The log files can then be used to update a standby database.

This command can only be executed when the invoking application or shell does not have a database connection to the specified database. This prevents a user from executing the command with uncommitted transactions. As such, the ARCHIVE LOG command will not forcibly commit the user's incomplete transactions. If the invoking application or shell already has a database connection to the specified database, the command will terminate and return an error. If another application has transactions in progress with the specified database when this command is executed, there will be a slight performance degradation since the command flushes the log buffer to disk. Any other transactions attempting to write log records to the buffer will have to wait until the flush is complete.

If used in a partitioned database environment, a subset of database partitions can be specified by using a database partition number clause. If the database partition number clause is not specified, the default behavior for this command is to close and archive the active log on all database partitions.

Using this command will use up a portion of the active log space due to the truncation of the active log file. The active log space will resume its previous size when the truncated log becomes inactive. Frequent use of this command can drastically reduce the amount of the active log space available for transactions.

### **Compatibilities:**

For compatibility with versions earlier than Version 8:

- The keyword `NODE` can be substituted for `DBPARTITIONNUM`.
- The keyword `NODES` can be substituted for `DBPARTITIONNUMS`.



**Related reference:**

- “db2ArchiveLog API - Archive the active log file” in *Administrative API Reference*

---

**ATTACH**

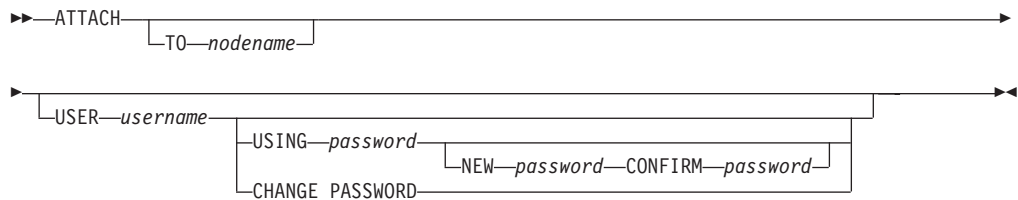
Enables an application to specify the instance at which instance-level commands (CREATE DATABASE and FORCE APPLICATION, for example) are to be executed. This instance can be the current instance, another instance on the same workstation, or an instance on a remote workstation.

**Authorization:**

None

**Required connection:**

None. This command establishes an instance attachment.

**Command syntax:****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 can 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. When attaching to a DB2 database instance on a Windows operating system, the user name can be specified in a format compatible with Microsoft Security Account Manager (SAM). The qualifier must be a NetBIOS-style name, which has a maximum length of 15 characters. For example, *domainname\username*.

**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.

**Related tasks:**

- “Attaching to and detaching from a non-default instance of the database manager” in *Administration Guide: Implementation*

**Related reference:**

- “DETACH ” on page 423
- “sqleatcp API - Attach to instance and change password” in *Administrative API Reference*
- “sqleatin API - Attach to instance” in *Administrative API Reference*

## AUTOCONFIGURE

Calculates and displays initial values for the buffer pool size, database configuration and database manager configuration parameters, with the option of applying these recommended values.

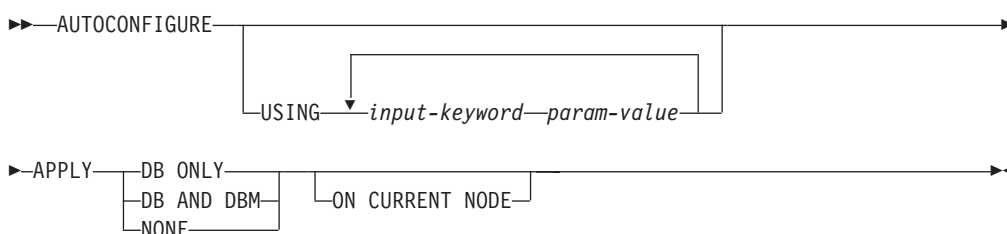
**Authorization:**

*sysadm.*

**Required connection:**

Database.

**Command syntax:**



**Command parameters:**

**USING input-keyword param-value**

Table 8. Valid input keywords and parameter values

| Keyword       | Valid values           | Default value | Explanation                                                                                                                                      |
|---------------|------------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| mem_percent   | 1-100                  | 25            | Percentage of memory to dedicate. If other applications (other than the operating system) are running on this server, set this to less than 100. |
| workload_type | simple, mixed, complex | mixed         | Simple workloads tend to be I/O intensive and mostly transactions, whereas complex workloads tend to be CPU intensive and mostly queries.        |
| num_stmts     | 1-1 000 000            | 10            | Number of statements per unit of work                                                                                                            |
| tpm           | 1-200 000              | 60            | Transactions per minute                                                                                                                          |

*Table 8. Valid input keywords and parameter values (continued)*

| <b>Keyword</b>  | <b>Valid values</b>         | <b>Default value</b> | <b>Explanation</b>                                                                                                                                                                                                                                                                                                                     |
|-----------------|-----------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| admin_priority  | performance, recovery, both | both                 | Optimize for better performance (more transactions per minute) or better recovery time                                                                                                                                                                                                                                                 |
| is_populated    | yes, no                     | yes                  | Is the database populated with data?                                                                                                                                                                                                                                                                                                   |
| num_local_apps  | 0-5 000                     | 0                    | Number of connected local applications                                                                                                                                                                                                                                                                                                 |
| num_remote_apps | 0-5 000                     | 10                   | Number of connected remote applications                                                                                                                                                                                                                                                                                                |
| isolation       | RR, RS, CS, UR              | RR                   | Maximum isolation level of applications connecting to this database (Repeatable Read, Read Stability, Cursor Stability, Uncommitted Read). It is only used to determine values of other configuration parameters. Nothing is set to restrict the applications to a particular isolation level and it is safe to use the default value. |
| bp_resizeable   | yes, no                     | yes                  | Are buffer pools resizeable?                                                                                                                                                                                                                                                                                                           |

**APPLY**

**DB ONLY**

Displays the recommended values for the database configuration and the buffer pool settings based on the current database manager configuration. Applies the recommended changes to the database configuration and the buffer pool settings.

**DB AND DBM**

Displays and applies the recommended changes to the database manager configuration, the database configuration, and the buffer pool settings.

**NONE**

Displays the recommended changes, but does not apply them.

**ON CURRENT NODE**

In the Database Partitioning Feature (DPF), the Configuration Advisor updates the database configuration on all nodes by default. Running with the "ON CURRENT NODE" option makes the advisor apply the recommended database configuration to the coordinator (connection) node only.

The bufferpool changes are always applied to the system catalogs. Thus, all nodes are affected. The "ON CURRENT NODE" option does not matter for bufferpool recommendations.

## AUTOCONFIGURE

### Usage notes:

- On systems with multiple logical partitions, the *mem\_percent* parameter refers to the percentage of memory that is to be used by all logical partitions. For example, if DB2 uses 25% of the memory on the system, specify 25% regardless of the number of logical partitions. The database configuration recommendations made, however, will be adjusted for one logical partition.
- This command makes configuration recommendations for the currently connected database, assuming that the database is the only active database on the system. If more than one database is active on the system, adjust the `>mem_percent` parameter to reflect the current database's share of memory. For example, if the DB2 database uses 80% of the system's memory and there are two active databases on the system that should share the resources equally, specify 40% (80% divided by 2 databases) for the parameter *mem\_percent*.
- When explicitly invoking the Configuration Advisor with the **AUTOCONFIGURE** command, the setting of the `DB2_ENABLE_AUTOCONFIG_DEFAULT` registry variable will be ignored.
- Running the **AUTOCONFIGURE** command on a database will recommend enablement of the Self Tuning Memory Manager. However, if you run the **AUTOCONFIGURE** command on a database in an instance where `SHEAPTHRES` is not zero, sort memory tuning (`SORTHEAP`) will not be enabled automatically. To enable sort memory tuning (`SORTHEAP`), you must set `SHEAPTHRES` equal to zero using the **UPDATE DATABASE MANAGER CONFIGURATION** command. Note that changing the value of `SHEAPTHRES` may affect the sort memory usage in your previously existing databases.

### Related concepts:

- "Configuration parameters" in *Performance Guide*

### Related tasks:

- "Defining the scope of configuration parameters using the Configuration Advisor" in *Administration Guide: Implementation*
- "Configuring DB2 with configuration parameters" in *Performance Guide*

### Related reference:

- "db2AutoConfig API - Access the Configuration Advisor" in *Administrative API Reference*
- "ADMIN\_CMD procedure – Run administrative commands" in *Administrative SQL Routines and Views*
- "AUTOCONFIGURE command using the ADMIN\_CMD procedure" in *Administrative SQL Routines and Views*

## BACKUP DATABASE

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

For information on the backup operations supported by DB2 database systems between different operating systems and hardware platforms, see "Backup and restore operations between different operating systems and hardware platforms" in the *Related concepts* section.

### Scope:

This command only affects the database partition on which it is executed.

### Authorization:

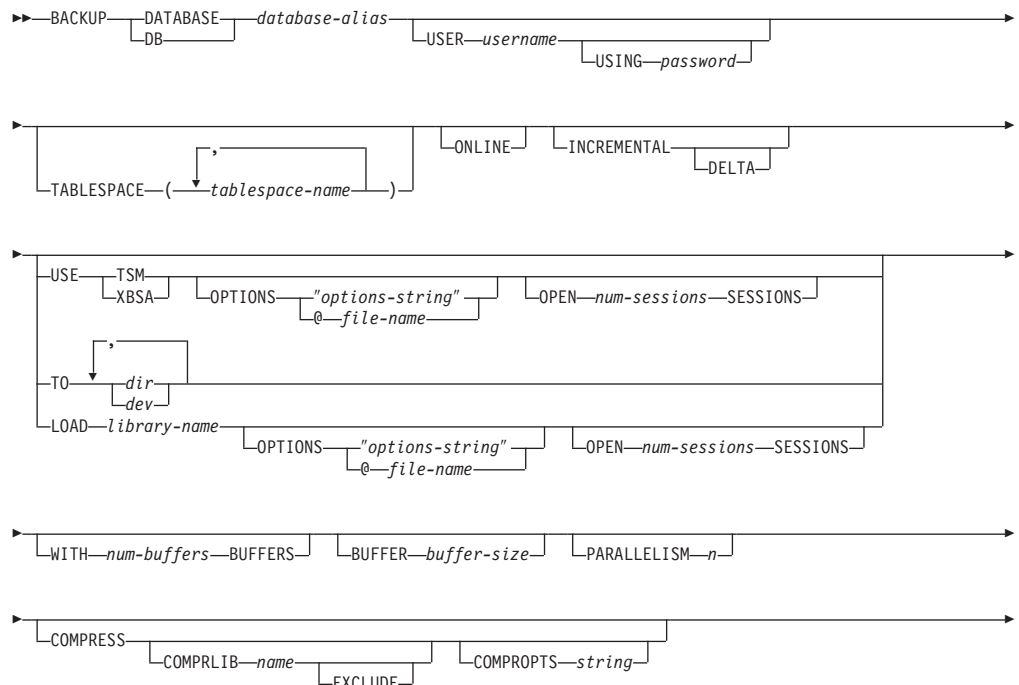
One of the following:

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

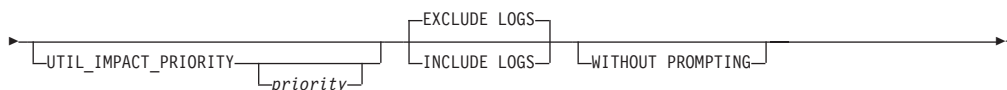
### Required connection:

Database. This command automatically establishes a connection to the specified database. If a connection to the specified database already exists, that connection will be terminated and a new connection established specifically for the backup operation. The connection is terminated at the completion of the backup operation.

### Command syntax:



## BACKUP DATABASE



### 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. During an online backup, DB2 obtains IN (Intent None) locks on all tables existing in SMS table spaces as they are processed and S (Share) locks on LOB data in SMS table spaces.

#### **INCREMENTAL**

Specifies a cumulative (incremental) backup image. An incremental backup image is a copy of all database data that has changed since the most recent successful, full backup operation.

#### **DELTA**

Specifies a non-cumulative (delta) backup image. A delta backup image is a copy of all database data that has changed since the most recent successful backup operation of any type.

#### **USE TSM**

Specifies that the backup is to use Tivoli Storage Manager output.

#### **USE XBSA**

Specifies that the XBSA interface is to be used. Backup Services APIs (XBSA) are an open application programming interface for applications or facilities needing data storage management for backup or archiving purposes.

#### **OPTIONS**

*"options-string"*

Specifies options to be used for the backup operation. The string will be passed to the vendor support library, for example TSM, exactly as it was entered, without the quotes. Specifying this option overrides the value specified by the `VENDOROPT` database configuration parameter.

*@file-name*

Specifies that the options to be used for the backup operation are contained in a file located on the DB2 server. The string will be passed to the vendor support library, for example TSM. The file must be a fully qualified file name.

#### **OPEN num-sessions SESSIONS**

The number of I/O sessions to be created between DB2 and TSM or



another backup vendor product. This parameter has no effect when backing up to tape, disk, or other local device.

### **TO dir/dev**

A list of directory or tape device names. The full path on which the directory resides must be specified. If USE TSM, TO, and LOAD are omitted, the default target directory for the backup image is the current working directory of the client computer. This target directory or device must exist on the database server. This parameter can 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 operation. Because there is no general tape support on Windows operating systems, each type of tape device requires a unique device driver. To back up to the FAT file system on Windows operating systems, users must conform to the 8.3 naming restriction.

Use of tape devices or floppy disks might 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 operation.

If the tape system does not support the ability to uniquely reference a backup image, it is recommended that multiple backup copies of the same database not be kept on the same tape.

### **LOAD library-name**

The name of the shared library (DLL on Windows operating systems) 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. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value. However, when creating a backup to multiple locations, a larger number of buffers can be used to improve performance.

### **BUFFER buffer-size**

The size, in 4 KB pages, of the buffer used when building the backup image. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value. The minimum value for this parameter is 8 pages.

If using tape with variable block size, reduce the buffer size to within the range that the tape device supports. Otherwise, the backup operation might succeed, but the resulting image might not be recoverable.

With most versions of Linux, using DB2's default buffer size for backup operations to a SCSI tape device results in error SQL2025N, reason code 75. To prevent the overflow of Linux internal SCSI buffers, use this formula:

$$\text{bufferpages} \leq \text{ST\_MAX\_BUFFERS} * \text{ST\_BUFFER\_BLOCKS} / 4$$

## BACKUP DATABASE

where *bufferpages* is the value you want to use with the BUFFER parameter, and ST\_MAX\_BUFFERS and ST\_BUFFER\_BLOCKS are defined in the Linux kernel under the drivers/scsi directory.

### PARALLELISM *n*

Determines the number of table spaces which can be read in parallel by the backup utility. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value.

### UTIL\_IMPACT\_PRIORITY *priority*

Specifies that the backup will run in throttled mode, with the priority specified. Throttling allows you to regulate the performance impact of the backup operation. Priority can be any number between 1 and 100, with 1 representing the lowest priority, and 100 representing the highest priority. If the UTIL\_IMPACT\_PRIORITY keyword is specified with no priority, the backup will run with the default priority of 50. If UTIL\_IMPACT\_PRIORITY is not specified, the backup will run in unthrottled mode. An impact policy must be defined by setting the *util\_impact\_lim* configuration parameter for a backup to run in throttled mode.

### COMPRESS

Indicates that the backup is to be compressed.

### COMPRLIB *name*

Indicates the name of the library to be used to perform the compression. The name must be a fully qualified path referring to a file on the server. If this parameter is not specified, the default DB2 compression library will be used. If the specified library cannot be loaded, the backup will fail.

### EXCLUDE

Indicates that the compression library will not be stored in the backup image.

### COMPROPTS *string*

Describes a block of binary data that will be passed to the initialization routine in the compression library. DB2 will pass this string directly from the client to the server, so any issues of byte reversal or code page conversion will have to be handled by the compression library. If the first character of the data block is '@', the remainder of the data will be interpreted by DB2 as the name of a file residing on the server. DB2 will then replace the contents of string with the contents of this file and will pass this new value to the initialization routine instead. The maximum length for *string* is 1024 bytes.

### EXCLUDE LOGS

Specifies that the backup image should not include any log files. When performing an offline backup operation, logs are excluded whether or not this option is specified.

### INCLUDE LOGS

Specifies that the backup image should include the range of log files required to restore and roll forward this image to some consistent point in time. This option is not valid for an offline backup.

### WITHOUT PROMPTING

Specifies that the backup will run unattended, and that any actions which normally require user intervention will return an error message.

### Examples:

1. In the following example, the database WSDB is defined on all 4 database partitions, numbered 0 through 3. The path /dev3/backup is accessible from all database partitions. Database partition 0 is the catalog partition, and needs to be backed-up separately since this is an offline backup. To perform an offline backup of all the WSDB database partitions to /dev3/backup, issue the following commands from one of the database partitions:

```
db2_all '<<+0< db2 BACKUP DATABASE wsdb TO /dev3/backup'
db2_all '|<<-0< db2 BACKUP DATABASE wsdb TO /dev3/backup'
```

In the second command, the db2\_all utility will issue the same backup command to each database partition in turn (except database partition 0). All four database partition backup images will be stored in the /dev3/backup directory.

2. In the following example database SAMPLE is backed up to a TSM server using two concurrent TSM client sessions. DB2 calculates the optimal buffer size for this environment.

```
db2 backup database sample use tsm open 2 sessions with 4 buffers
```

3. In the following example, a table space-level backup of table spaces (syscatspace, userspace1) of database payroll is done to tapes.

```
db2 backup database payroll tablespace (syscatspace, userspace1) to
/dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```

4. The USE TSM OPTIONS keywords can be used to specify the TSM information to use for the backup operation. The following example shows how to use the USE TSM OPTIONS keywords to specify a fully qualified file name:

```
db2 backup db sample use TSM options @/u/dmcinnis/myoptions.txt
```

The file myoptions.txt contains the following information: -fromnode=bar -fromowner=dmcinnis

5. Following is a sample weekly incremental backup strategy for a recoverable database. It includes a weekly full database backup operation, a daily non-cumulative (delta) backup operation, and a mid-week cumulative (incremental) backup operation:

```
(Sun) db2 backup db sample use tsm
(Mon) db2 backup db sample online incremental delta use tsm
(Tue) db2 backup db sample online incremental delta use tsm
(Wed) db2 backup db sample online incremental use tsm
(Thu) db2 backup db sample online incremental delta use tsm
(Fri) db2 backup db sample online incremental delta use tsm
(Sat) db2 backup db sample online incremental use tsm
```

6. In the following example, three identical target directories are specified for a backup operation on database SAMPLE. You might want to do this if the target file system is made up of multiple physical disks.

```
db2 backup database sample to /dev3/backup, /dev3/backup, /dev3/backup
```

The data will be concurrently backed up to the three target directories, and three backup images will be generated with extensions .001, .002, and .003.

### Usage notes:

The data in a backup cannot be protected by the database server. Make sure that backups are properly safeguarded, particularly if the backup contains LBAC-protected data.

## BACKUP DATABASE

When backing up to tape, use of a variable block size is currently not supported. If you must use this option, ensure that you have well tested procedures in place that enable you to recover successfully, using backup images that were created with a variable block size.

When using a variable block size, you must specify a backup buffer size that is less than or equal to the maximum limit for the tape devices that you are using. For optimal performance, the buffer size must be equal to the maximum block size limit of the device being used.

### **Related concepts:**

- “Backup and restore operations between different operating systems and hardware platforms” in *Data Recovery and High Availability Guide and Reference*
- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

### **Related tasks:**

- “Using backup” in *Data Recovery and High Availability Guide and Reference*

### **Related reference:**

- “BACKUP DATABASE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

# 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 database partition in `db2nodes.cfg`. It updates the database catalogs on the catalog database partition. Its effects are visible to all database partitions.

**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.

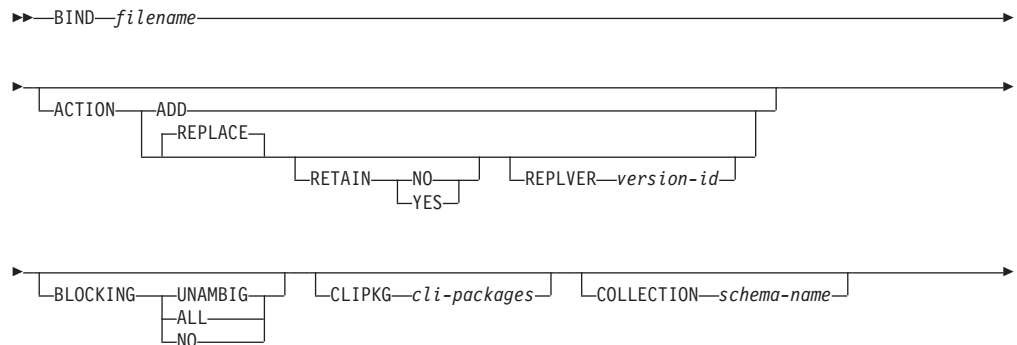
The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements. If the user has *sysadm* authority, but not explicit privileges to complete the bind, the database manager grants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

**Required connection:**

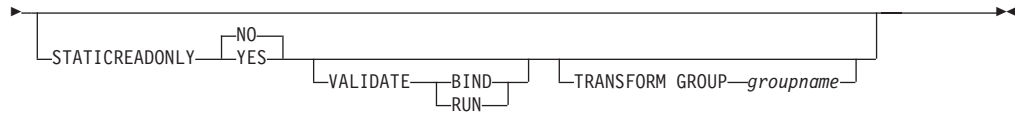
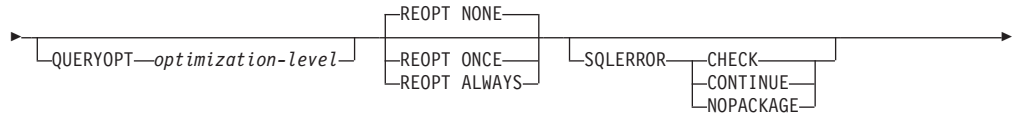
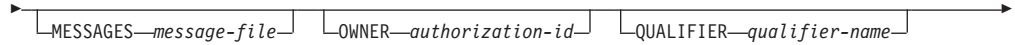
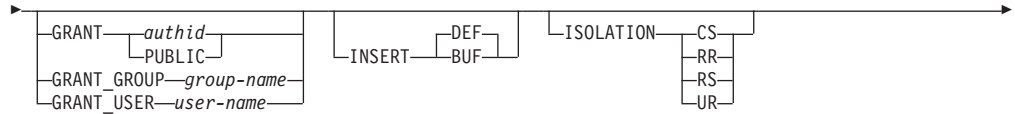
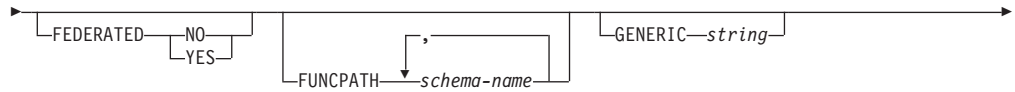
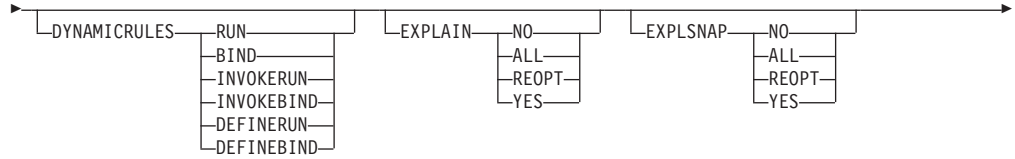
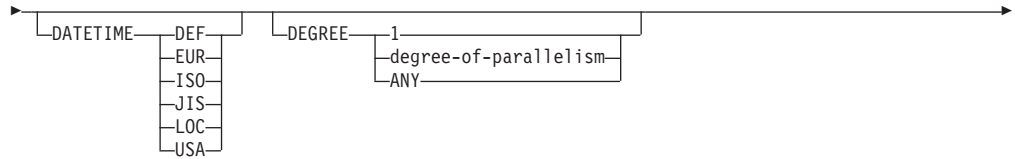
Database. If implicit connect is enabled, a connection to the default database is established.

**Command syntax:**

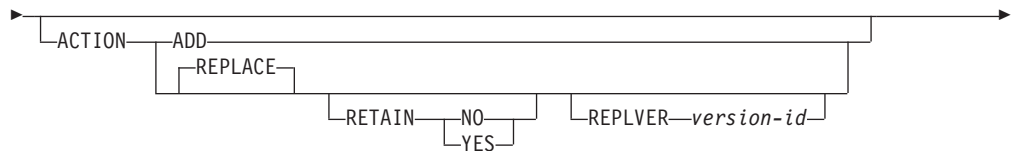
For DB2 for Windows and UNIX

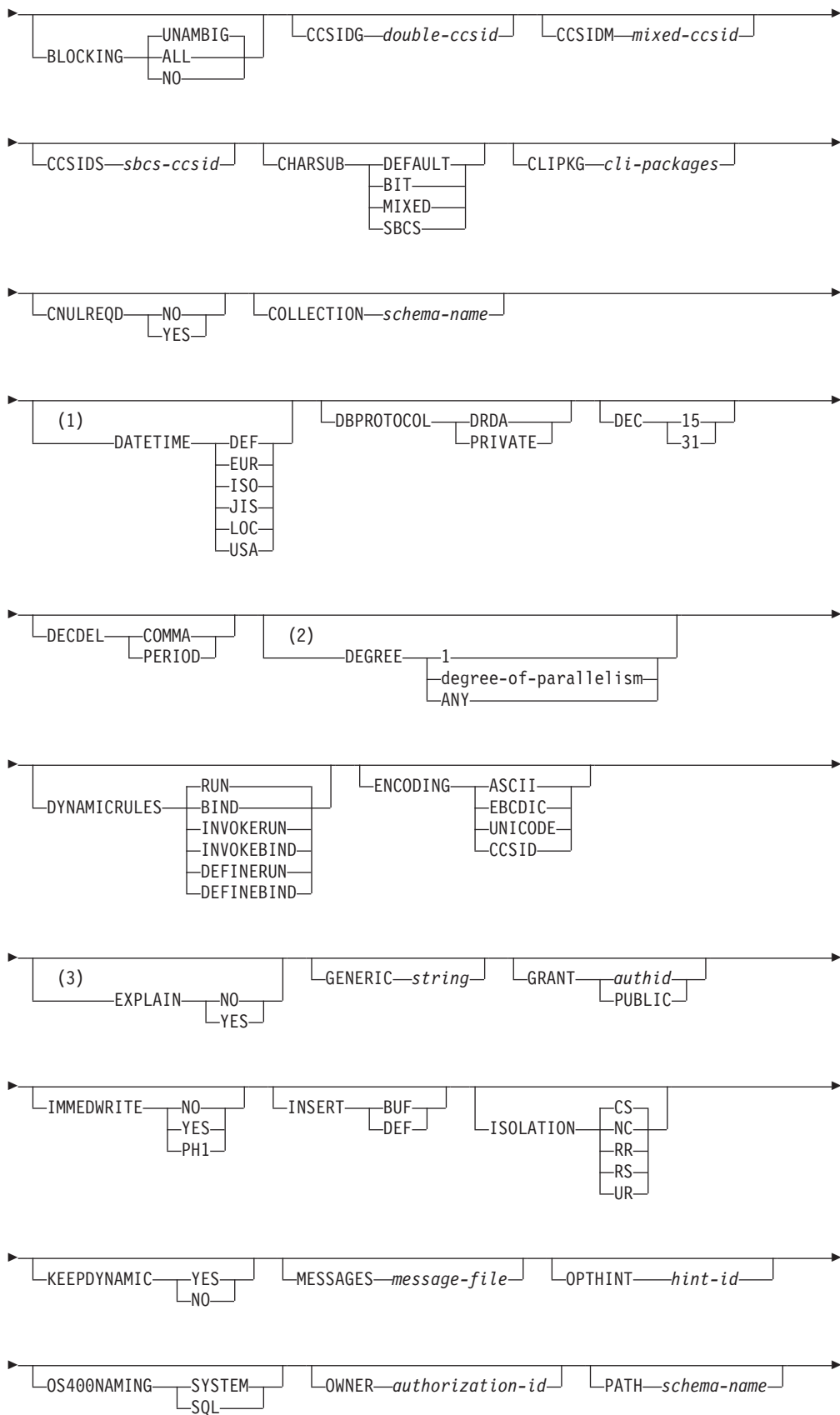


# BIND

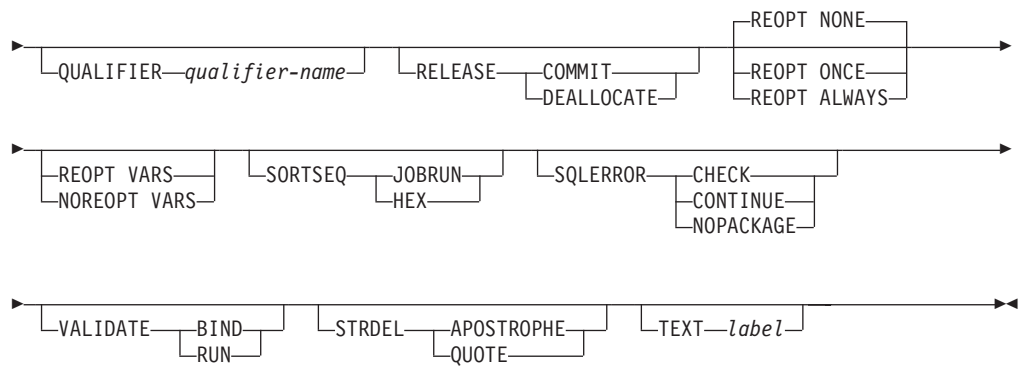


## For DB2 on servers other than Windows and UNIX





## BIND



### Notes:

- 1 If the server does not support the DATETIME DEF option, it is mapped to DATETIME ISO.
- 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 server does not support the EXPLAIN YES option, the value is mapped to EXPLAIN ALL.

### 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/sql11ib/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.

**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 existing package is to be replaced by a new one with the same package name and creator. This is the default value for the ACTION option.

### RETAIN

Indicates whether BIND and 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 BIND and EXECUTE authorities when a package is replaced. This value is not supported by DB2.

**YES** Preserves BIND and EXECUTE authorities when a package is replaced. This is the default value.

**REPLVER version-id**

Replaces a specific version of a package. The version identifier specifies which version of the package is to be replaced. If the specified version does not exist, an error is returned. If the REPLVER option of REPLACE is not specified, and a package already exists that matches the package name, creator, and version of the package being bound, that package will be replaced; if not, a new package will be added.

**BLOCKING**

Specifies the type of row blocking for cursors.

**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 option is not supported by DB2 Database for Linux, UNIX, and Windows. 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 option is not supported by DB2 Database for Linux, UNIX, and Windows. 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.

## BIND

This option is not supported by DB2 Database for Linux, UNIX, and Windows. 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 for Windows and UNIX.

**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.

### **CLIPKG cli-packages**

An integer between 3 and 30 specifying the number of CLI large packages to be created when binding CLI bind files against a database.

### **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 for Windows and UNIX.

**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 a 30-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.

**DEF** Use a date and time format associated with the territory 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 territory code of the database.

**USA** Use the IBM standard for U.S. date and time format.

**DBPROTOCOL**

Specifies what protocol to use when connecting to a remote site that is identified by a three-part name statement. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**DEC** Specifies the maximum precision to be used in decimal arithmetic operations. This DRDA precompile/bind option is not supported by DB2 for Windows and UNIX. 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 for Windows and UNIX. 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 for authorization checking of dynamic SQL statements. The authorization ID will also be used as the default package qualifier for implicit qualification of unqualified object references within dynamic SQL statements. 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.

**DEFINERUN**

If the package is used within a routine context, the authorization

## **BIND**

ID of the routine definer is to be used for authorization checking and for implicit qualification of unqualified object references within dynamic SQL statements within the routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with DYNAMICRULES RUN.

### **DEFINEBIND**

If the package is used within a routine context, the authorization ID of the routine definer is to be used for authorization checking and for implicit qualification of unqualified object references within dynamic SQL statements within the routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with DYNAMICRULES BIND.

### **INVOKERUN**

If the package is used within a routine context, the current statement authorization ID in effect when the routine is invoked is to be used for authorization checking of dynamic SQL statements and for implicit qualification of unqualified object references within dynamic SQL statements within that routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with DYNAMICRULES RUN.

### **INVOKEBIND**

If the package is used within a routine context, the current statement authorization ID in effect when the routine is invoked is to be used for authorization checking of dynamic SQL statements and for implicit qualification of unqualified object references within dynamic SQL statements within that routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with DYNAMICRULES BIND.

Because dynamic SQL statements will be using the authorization ID of the package owner in a package exhibiting bind behavior, the binder of the package should not have any authorities granted to them that the user of the package should not receive. Similarly, when defining a routine that will exhibit define behavior, the definer of the routine should not have any authorities granted to them that the user of the package should not receive since a dynamic statement will be using the authorization ID of the routine's definer.

The following dynamically prepared SQL statements cannot be used within a package that was not bound with DYNAMICRULES RUN: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET INTEGRITY, and SET EVENT MONITOR STATE.

### **ENCODING**

Specifies the encoding for all host variables in static statements in the plan or package. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**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 at prep/bind time for static statements and at run time for incremental bind statements.

If the package is to be used for a routine and the package contains incremental bind statements, then the routine must be defined as MODIFIES SQL DATA. If this is not done, incremental bind statements in the package will cause a run time error (SQLSTATE 42985).

**REOPT**

Explain information for each reoptimizable incremental bind SQL statement is placed in the explain tables at run time. In addition, explain information is gathered for reoptimizable dynamic SQL statements at run time, even if the CURRENT EXPLAIN MODE register is set to NO.

If the package is to be used for a routine, the routine must be defined as MODIFIES SQL DATA, otherwise incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985).

**ALL** Explain information for each eligible static SQL statement will be placed in the Explain tables at prep/bind time. Explain information for each eligible incremental bind SQL statement will be placed in the Explain tables at run time. In addition, Explain information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN MODE register is set to NO.

If the package is to be used for a routine, the routine must be defined as MODIFIES SQL DATA, otherwise incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985). 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 at prep/bind time for static statements and at run time for incremental bind statements.

If the package is to be used for a routine and the package contains incremental bind statements, then the routine must be defined as MODIFIES SQL DATA or incremental bind statements in the package will cause a run time error (SQLSTATE 42985).

**REOPT**

Explain snapshot information for each reoptimizable incremental bind SQL statement is placed in the explain tables at run time. In addition, explain snapshot information is gathered for

## BIND

reoptimizable dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT register is set to NO.

If the package is to be used for a routine, the routine must be defined as MODIFIES SQL DATA, otherwise incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985).

**ALL** An Explain Snapshot for each eligible static SQL statement will be placed in the Explain tables at prep/bind time. Explain snapshot information for each eligible incremental bind SQL statement will be placed in the Explain tables at run time. 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.

If the package is to be used for a routine, then the routine must be defined as MODIFIES SQL DATA, otherwise incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985).

### FEDERATED

Specifies whether a static SQL statement in a package references a nickname or a federated view. If this option is not specified and a static SQL statement in the package references a nickname or a federated view, a warning is returned and the package is created. This option is not supported for DRDA.

**NO** A nickname or federated view is not referenced in the static SQL statements of the package. If a nickname or federated view is encountered in a static SQL statement during the prepare or bind phase of this package, an error is returned and the package is *not* created.

**YES** A nickname or federated view can be referenced in the static SQL statements of the package. If no nicknames or federated views are encountered in static SQL statements during the prepare or bind of the package, no errors or warnings are returned and the package is created.

### 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.

#### schema-name

An 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.

### GENERIC string

Supports the binding of new options that are defined in the target database, but are not supported by DRDA. Do not use this option to pass

bind options that *are* defined in BIND or PRECOMPILE. 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 Universal Database, Version 8, one could use:

```
generic "explsnap all queryopt 3 federated yes"
```

to bind each of the EXPLSNAP, QUERYOPT, and FEDERATED options.

The maximum length of the string is 1023 bytes.

### **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.

If more than one of the GRANT, GRANT\_GROUP, and GRANT\_USER options are specified, only the last option specified is executed.

### **INSERT**

Allows a program being precompiled or bound against a DB2 Enterprise Server 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.

**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 for Windows and UNIX.

**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.

### **IMMEDWRITE**

Indicates whether immediate writes will be done for updates made to group buffer pool dependent pagesets or database partitions. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**KEEPDYNAMIC**

Specifies whether dynamic SQL statements are to be kept after commit points. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**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.

**OPTHINT**

Controls whether query optimization hints are used for static SQL. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**OS400NAMING**

Specifies which naming option is to be used when accessing DB2 UDB for iSeries data. Supported by DB2 UDB for iSeries only. For a list of supported option values, refer to the documentation for DB2 for iSeries.

Because of the slashes used as separators, a DB2 utility can still report a syntax error at execution time on certain SQL statements which use the iSeries system naming convention, even though the utility might have been precompiled or bound with the OS400NAMING SYSTEM option. For example, the Command Line Processor will report a syntax error on an SQL CALL statement if the iSeries system naming convention is used, whether or not it has been precompiled or bound using the OS400NAMING SYSTEM option.

**OWNER authorization-id**

Designates a 30-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.

**PATH** 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.

**schema-name**

An 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.

**QUALIFIER qualifier-name**

Provides a 30-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. The SET CURRENT



QUERY OPTIMIZATION statement describes the complete range of optimization levels available. 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 for Windows and UNIX.

#### COMMIT

Release resources at each COMMIT point. Used for dynamic SQL statements.

#### DEALLOCATE

Release resources only when the application terminates.

#### SORTSEQ

Specifies which sort sequence table to use on the iSeries system. Supported by DB2 UDB for iSeries only. For a list of supported option values, refer to the documentation for DB2 for iSeries.

#### 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

Creates a package, even if errors occur when binding SQL statements. Those statements that failed to bind for authorization or existence reasons can be incrementally bound at execution time if VALIDATE RUN is also specified. Any attempt to execute them at run time generates an error (SQLCODE -525, SQLSTATE 51015).

#### NOPACKAGE

A package or a bind file is not created if an error is encountered.

#### REOPT

Specifies whether to have DB2 determine an access path at run time using values for host variables, parameter markers, and special registers. Valid values are:

##### NONE

The access path for a given SQL statement containing host variables, parameter markers, or special registers will not be optimized using real values. The default estimates for these variables is used, and the plan is cached and will be used subsequently. This is the default value.

**ONCE** The access path for a given SQL statement will be optimized using the real values of the host variables, parameter markers, or special registers when the query is first executed. This plan is cached and used subsequently.

##### ALWAYS

The access path for a given SQL statement will always be compiled

and reoptimized using the values of the host variables, parameter markers, or special registers that are known each time the query is executed.

**REOPT / NOREOPT VARS**

These options have been replaced by REOPT ALWAYS and REOPT NONE; however, they are still supported for back-level compatibility. Specifies whether to have DB2 determine an access path at run time using values for host variables, parameter markers, and special registers. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**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).

**NO** Warnings will not be returned from the SQL compiler.

**YES** Warnings will be returned from the SQL compiler.

SQLCODE +236, +237 and +238 are exceptions. They are returned regardless of the **sqlwarn** option value.

**STATICREADONLY**

Determines whether static cursors will be treated as being READ ONLY. This DB2 precompile/bind option is not supported by DRDA.

**NO** All static cursors will take on the attributes as would normally be generated given the statement text and the setting of the LANGLEVEL precompile option. This is the default value.

**YES** Any static cursor that does not contain the FOR UPDATE or FOR READ ONLY clause will be considered READ ONLY.

**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 for Windows and UNIX. 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 for Windows and UNIX.

**TRANSFORM GROUP**

Specifies the transform group name to be used by static SQL statements for exchanging user-defined structured type values with host programs. This transform group is not used for dynamic SQL statements or for the exchange of parameters and results with external functions or methods. This option is not supported by DRDA.

**groupname**

An SQL identifier of up to 18 characters in length. A group name

cannot include a qualifier prefix and cannot begin with the prefix SYS since this is reserved for database use. In a static SQL statement that interacts with host variables, the name of the transform group to be used for exchanging values of a structured type is as follows:

- The group name in the TRANSFORM GROUP bind option, if any
- The group name in the TRANSFORM GROUP prep option as specified at the original precompilation time, if any
- The DB2\_PROGRAM group, if a transform exists for the given type whose group name is DB2\_PROGRAM
- No transform group is used if none of the above conditions exist.

The following errors are possible during the bind of a static SQL statement:

- SQLCODE yyyyyy, SQLSTATE xxxxx: A transform is needed, but no static transform group has been selected.
- SQLCODE yyyyyy, SQLSTATE xxxxx: The selected transform group does not include a necessary transform (TO SQL for input variables, FROM SQL for output variables) for the data type that needs to be exchanged.
- SQLCODE yyyyyy, SQLSTATE xxxxx: The result type of the FROM SQL transform is not compatible with the type of the output variable, or the parameter type of the TO SQL transform is not compatible with the type of the input variable.

In these error messages, *yyyyyy* is replaced by the SQL error code, and *xxxxx* by the SQL state code.

**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.

**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 can 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:

## BIND

```
db2 bind myapp.bnd
```

Any messages resulting from the bind process are sent to standard output.

### Usage notes:

Binding a package using the REOPT option with the ONCE or ALWAYS value specified might change the static and dynamic statement compilation and performance.

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.

When a package exhibits bind behavior, the following will be true:

1. The implicit or explicit value of the BIND option OWNER will be used for authorization checking of dynamic SQL statements.
2. The implicit or explicit value of the BIND option QUALIFIER will be used as the implicit qualifier for qualification of unqualified objects within dynamic SQL statements.
3. The value of the special register CURRENT SCHEMA has no effect on qualification.

In the event that multiple packages are referenced during a single connection, all dynamic SQL statements prepared by those packages will exhibit the behavior as specified by the DYNAMICRULES option for that specific package and the environment they are used in.

Parameters displayed in the SQL0020W message are correctly noted as errors, and will be ignored as indicated by the message.

If an SQL statement is found to be in error and the BIND option SQLERROR CONTINUE was specified, the statement will be marked as invalid. In order to change the state of the SQL statement, another BIND must be issued. Implicit and explicit rebind will not change the state of an invalid statement. In a package bound with VALIDATE RUN, a statement can change from static to incremental

bind or incremental bind to static across implicit and explicit rebinds depending on whether or not object existence or authority problems exist during the rebind.

**Related concepts:**

- “Isolation levels” in *SQL Reference, Volume 1*
- “Authorization considerations for dynamic SQL” in *Developing SQL and External Routines*
- “Effect of DYNAMICRULES bind option on dynamic SQL” in *Developing Embedded SQL Applications*
- “Performance improvements when using REOPT option of the BIND command” in *Developing Embedded SQL Applications*

**Related reference:**

- “PRECOMPILE ” on page 583
- “SET CURRENT QUERY OPTIMIZATION statement” in *SQL Reference, Volume 2*
- “Datetime values” in *SQL Reference, Volume 1*
- “Special registers” in *SQL Reference, Volume 1*
- “DB2 CLI bind files and package names” in *Call Level Interface Guide and Reference, Volume 1*
- “sqlabndx API - Bind application program to create a package” in *Administrative API Reference*

## 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 database partition on the server where the database resides.

### Authorization:

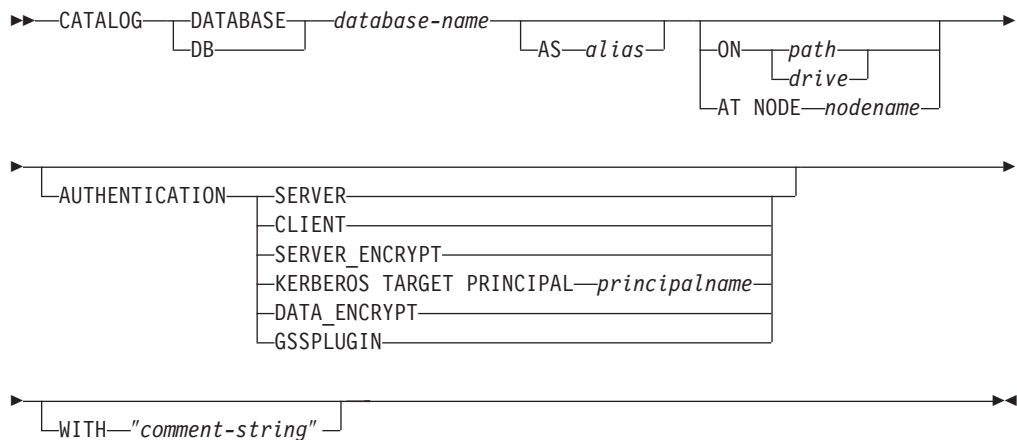
One of the following:

- *sysadm*
- *sysctrl*

### Required connection:

None. Directory operations affect the local directory only.

### Command syntax:



### 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*

Specifies the path on which the database being cataloged resides. On Windows operating systems, may instead specify the letter of the drive on which the database being cataloged resides (if it was created on a drive, not on a specific path).

#### **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

The authentication value is stored for remote databases (it appears in the output from the **LIST DATABASE DIRECTORY** command) but it is not stored for local databases.

Specifying an authentication type can result in a performance benefit.

### 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.

### 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.

### KERBEROS

Specifies that authentication takes place using Kerberos Security Mechanism. When authentication is Kerberos, only SECURITY=NONE is supported.

#### TARGET PRINCIPAL *principalname*

Fully qualified Kerberos principal name for the target server; that is, the fully qualified Kerberos principal of the DB2 instance owner in the form of *name/instance@REALM*. For Windows 2000, Windows XP, and Windows Server 2003, this is the logon account of the DB2 server service in the form of *userid@DOMAIN*, *userid@xxx.xxx.xxx.com* or *domain\userid*.

### DATA\_ENCRYPT

Specifies that authentication takes place on the node containing the target database, and that connections must use data encryption.

### GSSPLUGIN

Specifies that authentication takes place using an external GSS API-based plug-in security mechanism. When authentication is GSSPLUGIN, only SECURITY=NONE is supported.

### 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:

## CATALOG DATABASE

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 the LIST DATABASE DIRECTORY command. To list the contents of the local database directory use the LIST DATABASE DIRECTORY ON /PATH, where PATH is where the database was created.

If directory caching is enabled, database and node 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 might not be effective until the application has restarted.

To refresh the CLP's directory cache, use the TERMINATE command. 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.

### Related tasks:

- "Cataloging a database" in *Administration Guide: Implementation*

### Related reference:

- "sqlcadb API - Catalog a database in the system database directory" in *Administrative API Reference*
- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "LIST DATABASE DIRECTORY " on page 523
- "TERMINATE " on page 744
- "UNCATALOG DATABASE " on page 745



## CATALOG DCS DATABASE

Stores information about remote host or iSeries 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.

### Authorization:

One of the following:

- *sysadm*
- *sysctrl*

### Required connection:

None

### Command syntax:

```

▶▶ CATALOG DCS DATABASE database-name AS target-database-name
 DB
▶ AR library-name PARS "parameter-string"
▶ WITH "comment-string"

```

### Command parameters:

#### DATABASE database-name

Specifies the alias of the target database to catalog. This name should match the name of an entry in the database directory that is associated with the remote node.

#### AS target-database-name

Specifies the name of the target host or iSeries 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.

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 Windows operating systems, the path is `drive:\sql1lib\bin`. On UNIX based systems, the path is `$HOME/sql1lib/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. For more information on the parameter string, please refer to the "DCS directory values" topic through the **Related concepts** section.

## CATALOG DCS DATABASE

### 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 z/OS database, into the DCS directory:

```
db2 catalog dcs database db1 as dsn_db_1
with "DB2/z/OS location name DSN_DB_1"
```

### Usage notes:

The DB2 Connect program provides connections to DRDA Application Servers such as:

- DB2 for OS/390 or z/OS 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.
- iSeries databases on Application System/400 (AS/400) and iSeries 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 the LIST DCS DIRECTORY command.

If directory caching is enabled, 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 might not be effective until the application has restarted.

To refresh the CLP's directory cache, use the TERMINATE command. 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.

### Related concepts:

- "DCS directory values" in *DB2 Connect User's Guide*

### Related reference:

- "sqlgdad API - Catalog a database in the database connection services (DCS) directory" in *Administrative API Reference*
- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "TERMINATE " on page 744
- "UNCATALOG DCS DATABASE " on page 747
- "LIST DCS DIRECTORY " on page 531

**CATALOG LDAP DATABASE**

Used to register the database in Lightweight Directory Access Protocol (LDAP).

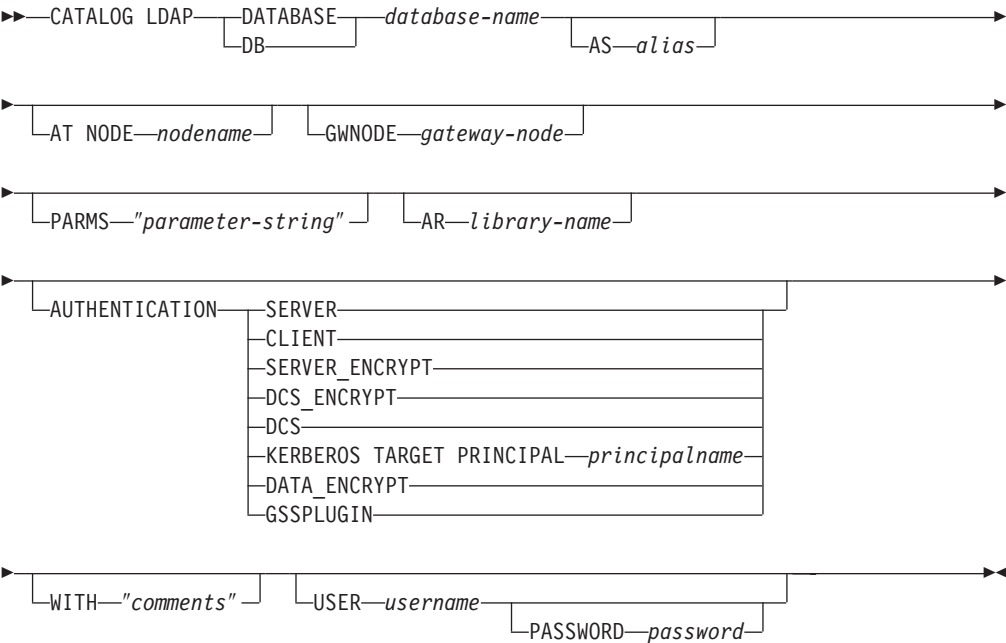
**Authorization:**

None

**Required connection:**

None

**Command syntax:**



**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. The change password *sym\_dest\_name*

## CATALOG LDAP DATABASE

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.

### **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.

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 Windows operating systems, the path is drive:\sql11b\d11. On UNIX based systems, the path is \$HOME/sql11b/lib of the instance owner.

## **AUTHENTICATION**

Specifies the authentication level. Valid values are:

### **SERVER**

Specifies that authentication takes place on the node containing the target database.

### **CLIENT**

Specifies that authentication takes place on the node from which the application is invoked.

### **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).

### **KERBEROS**

Specifies that authentication takes place using Kerberos Security Mechanism. When authentication is Kerberos, only SECURITY=NONE is supported.

#### **TARGET PRINCIPAL** *principalname*

Fully qualified Kerberos principal name for the target server; that is, the logon account of the DB2 server service in the form of *userid@xxx.xxx.xxx.com* or *domain\userid*.

### **DATA\_ENCRYPT**

Specifies that authentication takes place on the node containing the target database, and that connections must use data encryption.

### **GSSPLUGIN**

Specifies that authentication takes place using an external GSS API-based plug-in security mechanism. When authentication is GSSPLUGIN, only SECURITY=NONE is supported.

**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. If the user's LDAP DN and password have been specified using **db2ldcfg**, the user name and password do not have to be specified here.

**PASSWORD** *password*

Account password. If the user's LDAP DN and password have been specified using **db2ldcfg**, the user name and password do not have to be specified here.

**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.

It might 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 or iSeries database server. 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.

**Related concepts:**

- "Lightweight Directory Access Protocol (LDAP) overview" in *Administration Guide: Implementation*

**Related tasks:**

- "Registration of databases in the LDAP directory" in *Administration Guide: Implementation*

**Related reference:**

- "db2LdapCatalogDatabase API - Register the database on the LDAP server" in *Administrative API Reference*
- "REGISTER " on page 637
- "UNCATALOG LDAP DATABASE " on page 749

## CATALOG LDAP DATABASE

- "CATALOG LDAP NODE " on page 381
- "UNCATALOG LDAP NODE " on page 751
- "db2ldcfg - Configure LDAP environment " on page 140

## CATALOG LDAP NODE

Catalogs a new node entry in Lightweight Directory Access Protocol (LDAP).

### 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.

### Related concepts:

- "Lightweight Directory Access Protocol (LDAP) overview" in *Administration Guide: Implementation*

### Related tasks:

- "Registration of DB2 servers after installation" in *Administration Guide: Implementation*

### Related reference:

- "db2LdapCatalogNode API - Provide an alias for node name in LDAP server" in *Administrative API Reference*
- "CATALOG LDAP DATABASE " on page 377
- "UNCATALOG LDAP DATABASE " on page 749
- "UNCATALOG LDAP NODE " on page 751

## 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:**

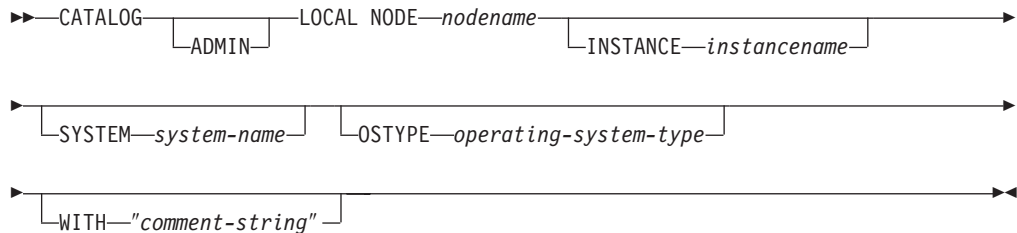
One of the following:

- *sysadm*
- *sysctrl*

**Required connection:**

None

**Command syntax:**



**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.

**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: AIX, WIN, HPUX, SUN, OS390, OS400, VM, VSE, SNI, SCO, LINUX and DYNIX.

**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:**

If directory caching is enabled, 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 might not be effective until the application has restarted.

To refresh the CLP's directory cache, use **TERMINATE**. 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.

**Related reference:**

- "sqlctnd API - Catalog an entry in the node directory" in *Administrative API Reference*
- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "TERMINATE " on page 744

## 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 only.

### Authorization:

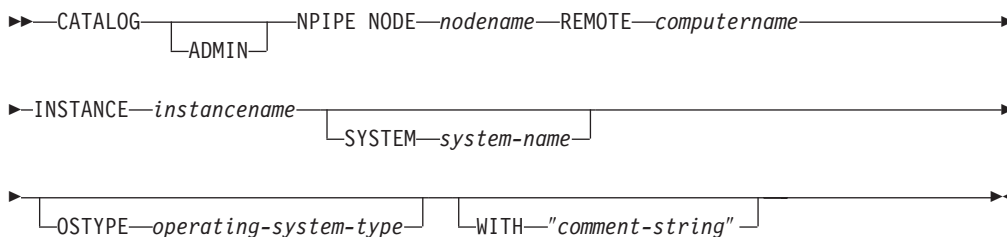
One of the following:

- *sysadm*
- *sysctrl*

### Required connection:

None

### Command syntax:



### 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.

#### 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: AIX, WIN, HPUX, SUN, OS390, OS400, VM, VSE, SNI, SCO, and LINUX.

#### 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 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 the LIST NODE DIRECTORY command.

If directory caching is enabled (see the configuration parameter *dir\_cache* in the GET DATABASE MANAGER CONFIGURATION command), 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 might not be effective until the application has restarted.

To refresh the CLP's directory cache, use the TERMINATE command. 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.

### Related reference:

- "sqlectnd API - Catalog an entry in the node directory" in *Administrative API Reference*
- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "LIST NODE DIRECTORY " on page 541
- "TERMINATE " on page 744

---

# 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. 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 platforms only.

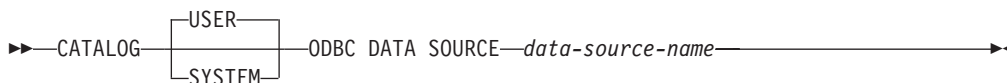
### Authorization:

None

### Required connection:

None

### Command syntax:



### 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.

### Related reference:

- "LIST ODBC DATA SOURCES " on page 544
- "UNCATALOG ODBC DATA SOURCE " on page 753

## CATALOG TCPIP/TCPIP4/TCPIP6 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. The **CATALOG TCPIP/TCPIP4/TCPIP6 NODE** command is run on a client.

### Authorization:

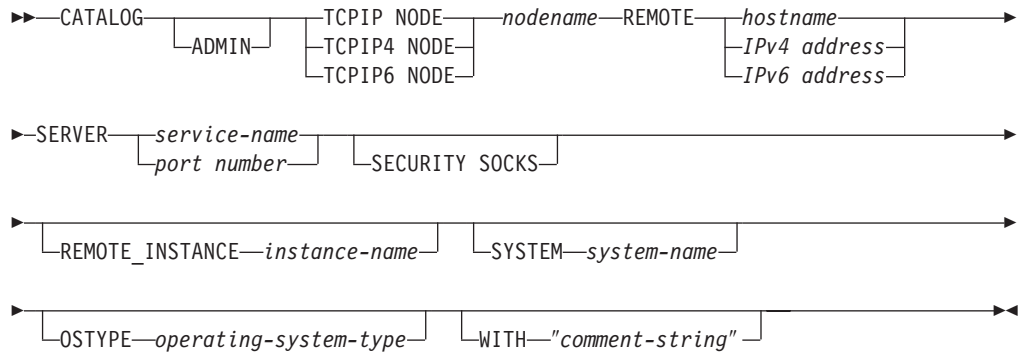
One of the following:

- *sysadm*
- *sysctrl*

### Required connection:

None. Directory operations affect the local directory only.

### Command syntax:



### Command parameters:

#### ADMIN

Specifies that a TCP/IP administration server node is to be cataloged. This parameter cannot be specified if the SECURITY SOCKS parameter is specified.

#### 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.

#### REMOTE *hostname/IPv4 address/IPv6 address*

The hostname or the IP address of the node where the target database resides. *IP address* can be an IPv4 or IPv6 address. The hostname is the name of the node that is known to the TCP/IP network. The maximum length of the hostname is 255 characters.

#### SERVER *service-name / port number*

Specifies the service name or the port number of the server database manager instance. The maximum length is 14 characters. This parameter is case sensitive.

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 server's database manager configuration file, and the *services* file on the

## CATALOG TCPIP/TCPIP4/TCPIP6 NODE

server is used to map this service name to a port number. The port number on the client and the server must match.

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, no service name needs to be specified in the local *services* file.

This parameter must not be specified for ADMIN nodes, but is mandatory for non-ADMIN nodes. The value on ADMIN nodes is always 523.

### SECURITY SOCKS

Specifies that the node will be SOCKS-enabled. This parameter is only supported for IPv4. If **CATALOG TCPIP NODE** is used and **SOCKS** is specified, the DB2 database product will use IPv4 to establish the connection. This parameter cannot be specified if the **ADMIN** parameter is specified.

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 a hostname or IPv4 address.

#### SOCKS\_SERVER

The fully qualified hostname or IPv4 address of the SOCKS server. If the SOCKSified DB2 client is unable to resolve the fully qualified hostname, it assumes that an IPv4 address has been entered.

One of the following conditions should be true:

- The SOCKS server is reachable via the domain name server.
- The hostname is listed in the hosts file. The location of this file is described in the TCP/IP documentation.
- An IPv4 address is specified.

If this command is issued after a **db2start**, it is necessary to issue a **TERMINATE** command to have the command take effect.

#### REMOTE\_INSTANCE *instance-name*

Specifies the name of the server instance to which an attachment or connection is being made.

#### SYSTEM *system-name*

Specifies the DB2 system name that is used to identify the server machine. This is the name of the physical machine, server system, or workstation.

#### OSTYPE *operating-system-type*

Specifies the operating system type of the server machine. Valid values are: AIX, WIN, HPUX, SUN, OS390, OS400, VM, VSE, and LINUX.

#### 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:

To specify a hostname using the **CATALOG TCPIP NODE** command, issue:

## CATALOG TCP/IP/TCP/IP4/TCP/IP6 NODE

```
db2 catalog tcpip node db2tcp1 remote hostname server db2inst1
with "Look up IPv4 or IPv6 address from hostname"
```

To specify an IPv4 address using the **CATALOG TCP/IP4 NODE** command, issue:

```
db2 catalog tcpip4 node db2tcp2 remote 192.0.32.67 server db2inst1
with "Look up IPv4 address from 192.0.32.67"
```

This example specifies an IPv4 address. You should not specify an IPv6 address in the **CATALOG TCP/IP4 NODE** command. The catalog will not fail if you do, but a subsequent attach or connect will fail because an invalid address was specified during cataloging.

To specify an IPv6 address using the **CATALOG TCP/IP6 NODE** command, issue:

```
db2 catalog tcpip6 node db2tcp3 1080:0:0:0:8:800:200C:417A server 50000
with "Look up IPv6 address from 1080:0:0:0:8:800:200C:417A"
```

This example specifies an IPv6 address and a port number for server. You should not specify an IPv6 address in the **CATALOG TCP/IP4 NODE** command. The catalog will not fail if you do, but a subsequent attach or connect will fail because an invalid address was specified during cataloging.

### 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 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 the **LIST NODE DIRECTORY** command.

If directory caching is enabled, 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 might not be effective until the application has restarted.

To refresh the CLP's directory cache, use the **TERMINATE** command. 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.

### Related reference:

- "sqlcnd API - Catalog an entry in the node directory" in *Administrative API Reference*
- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "LIST NODE DIRECTORY " on page 541
- "TERMINATE " on page 744

## 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 database partition on which it is executed.

**Authorization:**

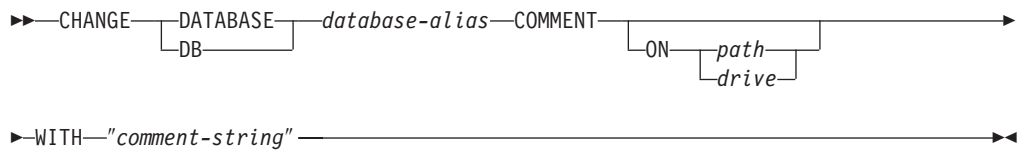
One of the following:

- *sysadm*
- *sysctrl*

**Required connection:**

None

**Command syntax:**



**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**

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 Windows operating systems, may instead specify the letter of the drive on which the database resides (if it was created on a drive, not on a specific path).

**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 info 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.

### Related reference:

- “sqlcsgd API - Change a database comment in the system or local database directory” in *Administrative API Reference*
- “CREATE DATABASE ” on page 395

---

## CHANGE ISOLATION LEVEL

Changes the way that DB2 isolates data from other processes while a database is being accessed.

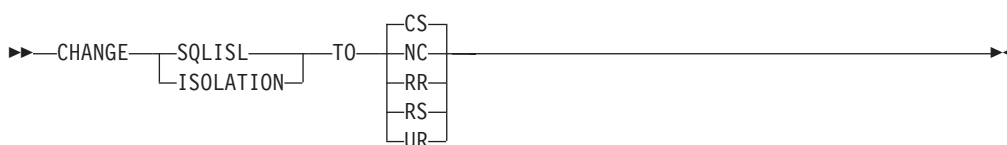
**Authorization:**

None

**Required connection:**

None

**Command syntax:**



**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. The back end process must be terminated before isolation level can be changed:

```
db2 terminate
db2 change isolation to ur
db2 connect to sample
```

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).

### **Related concepts:**

- "Isolation levels" in *SQL Reference, Volume 1*

### **Related reference:**

- "SET CLIENT " on page 716
- "QUERY CLIENT " on page 611
- "Change Isolation Level" in *Administrative API Reference*

## COMPLETE XMLSCHEMA

This command completes the process of registering an XML schema in the XML schema repository (XSR).

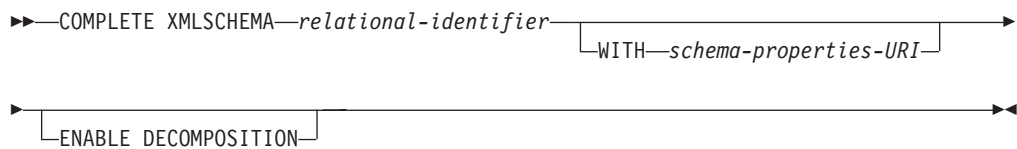
**Authorization:**

- The user ID must be the owner of the XSR object as recorded in the catalog view SYSCAT.XSROBJECTS.

**Required connection:**

Database

**Command syntax:**



**Description:**

*relational-identifier*

Specifies the relational name of an XML schema previously registered with the REGISTER XMLSCHEMA command. The relational name can be specified as a two-part SQL identifier, consisting of the SQL schema and the XML schema name, having the following format: *SQLschema.name*. The default SQL schema, as defined in the CURRENT SCHEMA special register, is used if no schema is specified.

**WITH *schema-properties-URI***

Specifies the uniform resource identifier (URI) of a properties document for the XML schema. Only a local file, specified by a file scheme URI, is supported. A schema property document can only be specified during the completion stage of XML schema registration.

**ENABLE DECOMPOSITION**

Indicates that the schema can be used for decomposing XML instance documents.

**Example:**

```
COMPLETE XMLSCHEMA user1.P0schema WITH 'file:///c:/TEMP/schemaProp.xml'
```

**Usage notes:**

An XML schema cannot be referenced or used for validation or annotation until the XML schema registration process has been completed. This command completes the XML schema registration process for an XML schema that was begun with the REGISTER XMLSCHEMA command.

**Related reference:**

- “ADD XMLSCHEMA DOCUMENT ” on page 339
- “REGISTER XMLSCHEMA ” on page 640

## CREATE DATABASE

The **CREATE DATABASE** command 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 file. When you initialize a new database, the **AUTOCONFIGURE** command is issued by default.

When the **CREATE DATABASE** command is issued, the Configuration Advisor also runs automatically. This means that the database configuration parameters are automatically tuned for you according to your system resources. In addition, Automated Runstats is enabled. To disable the Configuration Advisor from running at database creation, please refer to the *db2\_enable\_autoconfig\_default* registry variable. To disable Automated Runstats, please refer to *auto\_runstats* database configuration parameter.

Adaptive Self Tuning Memory is also enabled by default for single partition databases. To disable Adaptive Self Tuning Memory by default, refer to the *self\_tuning\_mem* database configuration parameter (see the *Related reference* section). For multi-partition databases, Adaptive Self Tuning Memory is disabled by default.

In order to make use of XML features, the codeset must be set to UTF-8 through the **USING CODESET** option of this command. In a future release of the DB2 database system, the default code set will be changed to UTF-8 when creating a database. If a particular code set and territory is needed for a database, the desired code set and territory should be specified in the **CREATE DATABASE** command

This command is not valid on a client.

### Scope:

In a partitioned database environment, this command affects all database partitions that are listed in the *db2nodes.cfg* file.

The database partition from which this command is issued becomes the catalog database partition for the new database.

### Authorization:

You must have one of the following:

- *sysadm*
- *sysctrl*

### Required connection:

Instance. To create a database at another (remote) node, you must first attach to that node. A database connection is temporarily established by this command during processing.

### Command syntax:

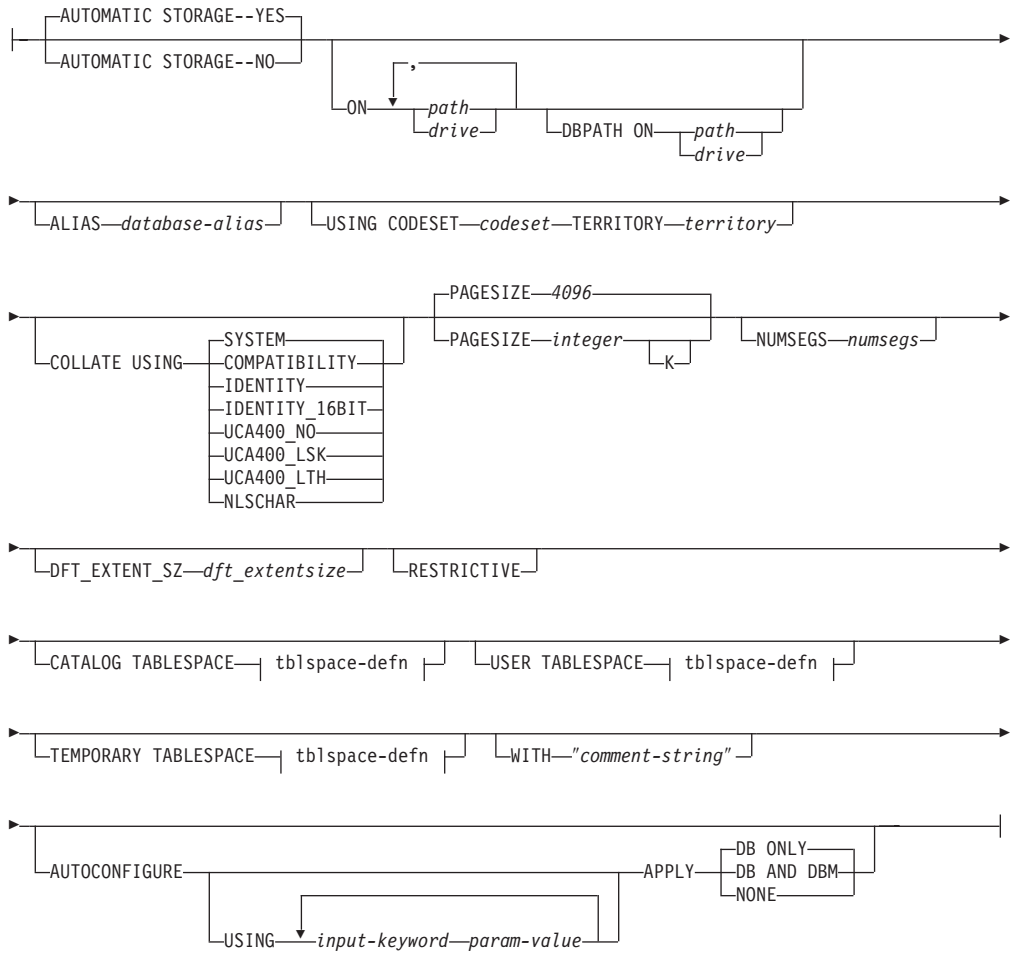
```

▶▶ CREATE DATABASE database-name
 └──┬── DB
 └──┬── AT DBPARTITIONNUM
 └── Create Database options

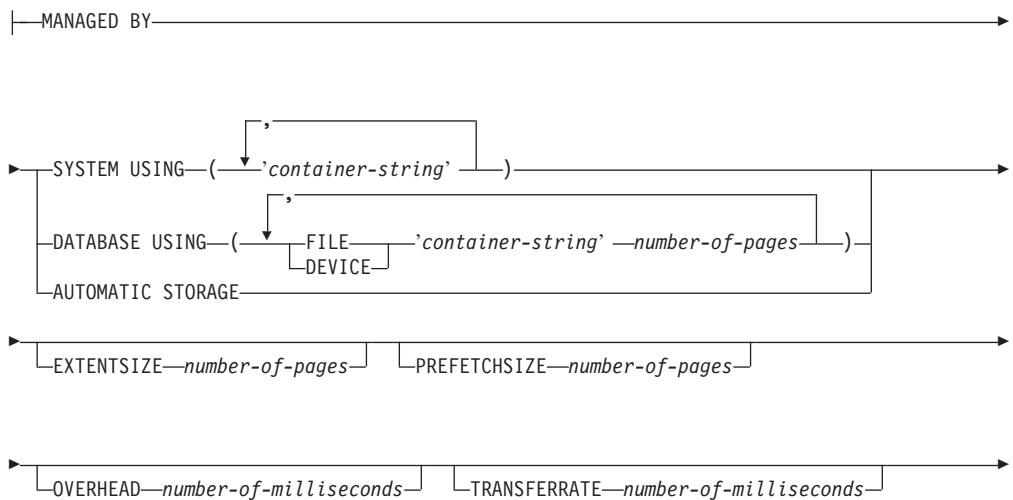
```

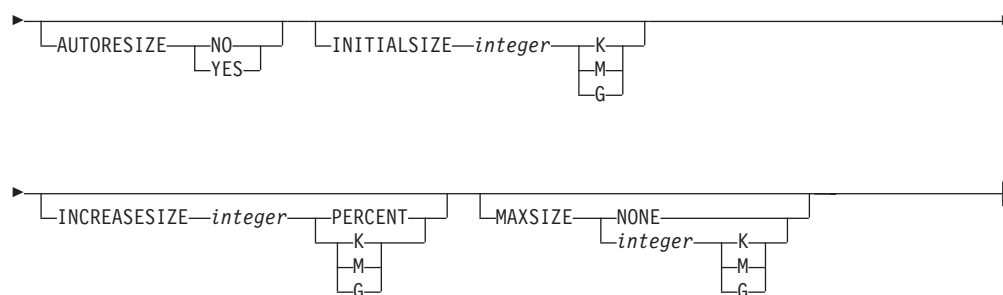
# CREATE DATABASE

## Create Database options:



## tblspace-defn:



**Notes:**

1. The combination of the code set and territory values must be valid.
2. Not all collating sequences are valid with every code set and territory combination.
3. The table space definitions specified on CREATE DATABASE apply to all database partitions on which the database is being created. They cannot be specified separately for each database partition. If the table space definitions are to be created differently on particular database partitions, the CREATE TABLESPACE statement must be used.

When defining containers for table spaces, \$N can be used. \$N will be replaced by the database partition number when the container is actually created. This is required if the user wants to specify containers in a multiple logical partition database.

4. The AUTOCONFIGURE option requires *sysadm* authority.

**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. Specifically, the name must not contain any space characters.

**AT DBPARTITIONNUM**

Specifies that the database is to be created only on the database partition that issues the command. You do not specify this option when you create a new database. You can use it to recreate a database partition that you dropped because it was damaged. After you use the CREATE DATABASE command with the AT DBPARTITIONNUM option, the database at this database partition is in the restore-pending state. You must immediately restore the database on this node. This parameter is not intended for general use. For example, it should be used with RESTORE DATABASE command 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.

If this parameter is used to recreate a database partition that was dropped (because it was damaged), the database at this database partition will be in the restore-pending state. After recreating the database partition, the database must immediately be restored on this database partition.

**AUTOMATIC STORAGE NO | YES**

Specifies that automatic storage is being explicitly disabled or enabled for

## CREATE DATABASE

the database. The default value is YES. If the AUTOMATIC STORAGE clause is not specified, automatic storage is implicitly enabled by default.

**NO** Automatic storage is not being enabled for the database.

**YES** Automatic storage is being enabled for the database.

### ON path or drive

The meaning of this option depends on the value of the AUTOMATIC STORAGE option.

- If AUTOMATIC STORAGE NO is specified, automatic storage is disabled for the database. In this case, only one path can be included as part of the ON option, and it specifies the path on which to create the database. If a path is not specified, the database is created on the default database path that is specified in the database manager configuration file (*dftdbpath* parameter). This behavior matches that of DB2 UDB Version 8.2 and earlier.

- Otherwise, automatic storage is enabled for the database by default. In this case, multiple paths may be listed here, each separated by a comma. These are referred to as storage paths and are used to hold table space containers for automatic storage table spaces. For multi-partition databases the same storage paths will be used on all partitions.

With multiple paths, the DBPATH ON option specifies which of the multiple paths on which to create the database. If the DBPATH ON option is not specified, the database is created on the first path listed. If no paths are specified, the database is created on the default database path that is specified in the database manager configuration file (*dftdbpath* parameter). This will also be used as the location for the single storage path associated with the database.

The maximum length of a path is 175 characters.

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. Also, all paths specified as part of the ON option must exist on all database partitions.

A given database path or storage path must exist and be accessible on each database partition.

### DBPATH ON path or drive

If automatic storage is enabled, the DBPATH ON option specifies the path on which to create the database. If automatic storage is enabled and the DBPATH ON option is not specified, the database is created on the first path listed with the ON option.

The maximum length of a database path is 215 characters and the maximum length of a storage path is 175 characters.

### 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. After you create the database, you cannot change the specified code set. To make use of XML features, the codeset must be set to UTF-8.



**TERRITORY territory**

Specifies the territory to be used for data entered into this database. After you create the database, you cannot change the specified territory.

**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. This is the default for Unicode databases.

**IDENTITY\_16BIT**

CESU-8 (Compatibility Encoding Scheme for UTF-16: 8-Bit) collation sequence as specified by the Unicode Technical Report #26, which is available at the Unicode Consortium Web site ([www.unicode.org](http://www.unicode.org)). This option can only be specified when creating a Unicode database.

**UCA400\_NO**

The UCA (Unicode Collation Algorithm) collation sequence that is based on the Unicode Standard version 4.00 with normalization implicitly set to on. Details of the UCA can be found in the Unicode Technical Standard #10, which is available at the Unicode Consortium Web site ([www.unicode.org](http://www.unicode.org)). This is the default collation when codeset UTF-8 is specified and can only be used when creating a Unicode database.

**UCA400\_LSK**

The UCA (Unicode Collation Algorithm) collation sequence based on the Unicode Standard version 4.00 but will sort Slovakian characters in the appropriate order. Details of the UCA can be found in the Unicode Technical Standard #10, which is available at the Unicode Consortium Web site ([www.unicode.org](http://www.unicode.org)). This option can only be used when creating a Unicode database.

**UCA400\_LTH**

The UCA (Unicode Collation Algorithm) collation sequence that is based on the Unicode Standard version 4.00 but will sort all Thai characters according to the Royal Thai Dictionary order. Details of the UCA can be found in the Unicode Technical Standard #10 available at the Unicode Consortium Web site ([www.unicode.org](http://www.unicode.org)). This option can only be used when creating a Unicode database. This collator might order Thai data differently from the NLSCHAR collator option.

**NLSCHAR**

System-defined collating sequence using the unique collation rules for the specific code set/territory.

This option can only be used with the Thai code page (CP874). If this option is specified in non-Thai environments, the command will fail and return the error SQL1083N with Reason Code 4.

**SYSTEM**

For non-Unicode databases, this is the default option, with the

## CREATE DATABASE

collating sequence based on the database territory. For Unicode databases, this option is equivalent to the IDENTITY option.

### **PAGESIZE integer**

Specifies the page size of the default buffer pool along with the initial table spaces (SYSCATSPACE, TEMPSPACE1, USERSPACE1) when the database is created. This also represents the default page size for all future CREATE BUFFERPOOL and CREATE TABLESPACE statements. The valid values for integer without the suffix K are 4 096, 8 192, 16 384, or 32 768. The valid values for integer with the suffix K are 4, 8, 16, or 32. At least one space is required between the integer and the suffix K. The default is a page size of 4 096 bytes (4 K).

### **NUMSEGS numsegs**

Specifies the number of directories (tablespace containers) that will be created and used to store the database table files for any default SMS table spaces. This parameter does not affect automatic storage table spaces, 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.

### **RESTRICTIVE**

If the RESTRICTIVE option is present it causes the RESTRICT\_ACCESS database configuration parameter to be set to YES and no privileges are automatically granted to PUBLIC. If the RESTRICTIVE option is not present then the RESTRICT\_ACCESS database configuration parameter is set to NO and all of the following privileges are automatically granted to PUBLIC.

- CREATETAB
- BINDADD
- CONNECT
- IMPLSCHEMA
- EXECUTE with GRANT on all procedures in schema SQLJ
- EXECUTE with GRANT on all functions and procedures in schema SYSPROC
- BIND on all packages created in the NULLID schema
- EXECUTE on all packages created in the NULLID schema
- CREATEIN on schema SQLJ
- CREATEIN on schema NULLID
- USE on table space USERSPACE1
- SELECT access to the SYSIBM catalog tables
- SELECT access to the SYSCAT catalog views
- SELECT access to the SYSSTAT catalog views
- UPDATE access to the SYSSTAT catalog views

### **CATALOG TABLESPACE tblspace-defn**

Specifies the definition of the table space that will hold the catalog tables, SYSCATSPACE. If not specified and automatic storage is not enabled for the database, SYSCATSPACE is 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

```

If not specified and automatic storage is enabled for the database, SYSCATSPACE is created as an automatic storage table space with its containers created on the defined storage paths. The extent size of this table space is 4. Appropriate values for AUTORESIZE, INITIALSIZE, INCREASESIZE, and MAXSIZE are set automatically.

See the CREATE TABLESPACE statement for more information on the table space definition fields.

In a partitioned database environment, the catalog table space is only created on the catalog database partition, the database partition 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 and automatic storage is not enabled for the database, USERSPACE1 is 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/SQLT0002.1
/u/smith/smith/NODE0000/SQL00001/SQLT0002.2
/u/smith/smith/NODE0000/SQL00001/SQLT0002.3
/u/smith/smith/NODE0000/SQL00001/SQLT0002.4

```

If not specified and automatic storage is enabled for the database, USERSPACE1 is created as an automatic storage table space with its containers created on the defined storage paths. The extent size of this table space will be *dft\_extentsize*. Appropriate values for AUTORESIZE, INITIALSIZE, INCREASESIZE, and MAXSIZE are set automatically.

See the CREATE TABLESPACE statement for more information on the table space definition fields.

#### TEMPORARY TABLESPACE *tblspace-defn*

Specifies the definition of the initial system temporary table space, TEMPSPACE1. If not specified and automatic storage is not enabled for the database, TEMPSPACE1 is 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/SQLT0001.1
/u/smith/smith/NODE0000/SQL00001/SQLT0001.2
/u/smith/smith/NODE0000/SQL00001/SQLT0001.3
/u/smith/smith/NODE0000/SQL00001/SQLT0001.4

```

If not specified and automatic storage is enabled for the database, TEMPSPACE1 is created as an automatic storage table space with its containers created on the defined storage paths. The extent size of this table space is *dft\_extentsize*.

## CREATE DATABASE

See the CREATE TABLESPACE statement for more information on the table space definition fields.

### 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.

### AUTOCONFIGURE

Based on user input, calculates the recommended settings for buffer pool size, database configuration, and database manager configuration and optionally applies them. The Configuration Advisor is run by default when the CREATE DATABASE command is issued. The AUTOCONFIGURE option is needed only if you want to tweak the recommendations.

### USING *input-keyword param-value*

Table 9. Valid input keywords and parameter values

| Keyword         | Valid values                | Default value | Explanation                                                                                                                                      |
|-----------------|-----------------------------|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| mem_percent     | 1–100                       | 25            | Percentage of memory to dedicate. If other applications (other than the operating system) are running on this server, set this to less than 100. |
| workload_type   | simple, mixed, complex      | mixed         | Simple workloads tend to be I/O intensive and mostly transactions, whereas complex workloads tend to be CPU intensive and mostly queries.        |
| num_stmts       | 1–1 000 000                 | 25            | Number of statements per unit of work                                                                                                            |
| tpm             | 1–200 000                   | 60            | Transactions per minute                                                                                                                          |
| admin_priority  | performance, recovery, both | both          | Optimize for better performance (more transactions per minute) or better recovery time                                                           |
| num_local_apps  | 0–5 000                     | 0             | Number of connected local applications                                                                                                           |
| num_remote_apps | 0–5 000                     | 100           | Number of connected remote applications                                                                                                          |
| isolation       | RR, RS, CS, UR              | RR            | Isolation level of applications connecting to this database (Repeatable Read, Read Stability, Cursor Stability, Uncommitted Read)                |

Table 9. Valid input keywords and parameter values (continued)

| Keyword       | Valid values | Default value | Explanation                  |
|---------------|--------------|---------------|------------------------------|
| bp_resizeable | yes, no      | yes           | Are buffer pools resizeable? |

**APPLY****DB ONLY**

Displays the recommended values for the database configuration and the buffer pool settings based on the current database manager configuration. Applies the recommended changes to the database configuration and the buffer pool settings.

**DB AND DBM**

Displays and applies the recommended changes to the database manager configuration, the database configuration, and the buffer pool settings.

**NONE**

Disables the Configuration Advisor (it is enabled by default).

- If the **AUTOCONFIGURE** keyword is specified with the **CREATE DATABASE** command, the **DB2\_ENABLE\_AUTOCONFIG\_DEFAULT** variable value is not considered. Adaptive Self Tuning Memory and Auto Runstats will be enabled and the Configuration Advisor will tune the database configuration and database manager configuration parameters as indicated by the **APPLY DB** or **APPLY DBM** options.
- Specifying the **AUTOCONFIGURE** option with the **CREATE DATABASE** command on a database will recommend enablement of the Self Tuning Memory Manager. However, if you run the **AUTOCONFIGURE** command on a database in an instance where **SHEAPTHRES** is not zero, sort memory tuning (**SORTHEAP**) will not be enabled automatically. To enable sort memory tuning (**SORTHEAP**), you must set **SHEAPTHRES** equal to zero using the **UPDATE DATABASE MANAGER CONFIGURATION** command. Note that changing the value of **SHEAPTHRES** may affect the sort memory usage in your previously existing databases.

**Usage notes:**

The **CREATE DATABASE** command:

- Creates a database in the specified subdirectory. In a partitioned database environment, creates the database on all database partitions listed in **db2nodes.cfg**, and creates a **\$DB2INSTANCE/NODExxxx** directory under the specified subdirectory at each database partition. In a single partition database environment, 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 by the *dftdbpath* parameter. 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.

## CREATE DATABASE

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 schemas called SYSCAT, SYSFUN, SYSIBM, and SYSSTAT with SYSIBM as the owner. The database partition server on which this command is issued becomes the catalog database partition for the new database. Two database partition groups are created automatically: IBMDEFAULTGROUP and IBMCATGROUP.
- 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.

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 database partition.
- Grants the following:
  - EXECUTE WITH GRANT privilege to PUBLIC on all functions in the SYSFUN schema
  - EXECUTE privilege to PUBLIC on all procedures in SYSIBM schema
  - DBADM authority, and CONNECT, CREATETAB, BINDADD, CREATE\_NOT\_FENCED, IMPLICIT\_SCHEMA and LOAD 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.
  - EXECUTE WITH GRANT privilege to PUBLIC on all functions in the SYSFUN schema.
  - EXECUTE privilege to PUBLIC on all procedures in SYSIBM schema.

Automatic storage is a collection of storage paths associated with a database on which table spaces can be created without having to explicitly specify container definitions (see the CREATE TABLESPACE statement for more information). Automatic storage is enabled by default, but can be explicitly disabled for a database when it is created. Automatic storage can be disabled at database creation time by specifying the AUTOMATIC STORAGE NO option.

It is important to note that automatic storage can only be enabled at database creation time, it cannot be enabled after the database has been created. Also, automatic storage cannot be disabled once a database has been defined to use it.

When free space is calculated for an automatic storage path for a given database partition, the database manager will check for the existence of the following directories or mount points within the storage path and will use the first one that is found. In doing this, file systems can be mounted at a point beneath the storage path and the database manager will recognize that the actual amount of free space available for table space containers may not be the same amount that is associated with the storage path directory itself.

1. <storage path>/<instance name>/NODE####/<database name>
2. <storage path>/<instance name>/NODE####
3. <storage path>/<instance name>
4. <storage path>/<

Where

- <storage path> is a storage path associated with the database.
- <instance name> is the instance under which the database resides.
- NODE#### corresponds to the database partition number (for example NODE0000 or NODE0001).
- <database name> is the name of the database.

Consider the example where two logical database partitions exist on one physical machine and the database is being created with a single storage path: /db2data. Each database partition will use this storage path but the user may wish to isolate the data from each partition within its own file system. In this case, a separate file system can be created for each partition and be mounted at /db2data/<instance>/NODE####. When creating containers on the storage path and determining free space, the database manager will know not to retrieve free space information for /db2data, but instead retrieve it for the corresponding /db2data/<instance>/NODE#### directory.

In general, the same storage paths must be used for each partition in a multi-partition database and they must all exist prior to executing the **CREATE DATABASE** command. One exception to this is where database partition expressions are used within the storage path. Doing this allows the database partition number to be reflected in the storage path such that the resulting path name is different on each partition.

You use the argument " \$N" ([blank]\$N) to indicate a database partition expression. A database partition expression can be used anywhere in the storage path, and multiple database partition expressions can be specified. Terminate the database partition expression with a space character; whatever follows the space is appended to the storage path after the database partition expression is evaluated. If there is no space character in the storage path after the database partition expression, it is assumed that the rest of the string is part of the expression. The argument can only be used in one of the following forms:

Operators are evaluated from left to right. % represents the modulus operator. The database partition number in the examples is assumed to be 10.

| Syntax     | Example | Value |
|------------|---------|-------|
| [blank]\$N | " \$N"  | 10    |

## CREATE DATABASE

Operators are evaluated from left to right. % represents the modulus operator. The database partition number in the examples is assumed to be 10.

| Syntax                       | Example    | Value |
|------------------------------|------------|-------|
| [blank]\$N+[number]          | " \$N+100" | 110   |
| [blank]\$N%[number]          | " \$N%5"   | 0     |
| [blank]\$N+[number]%[number] | " \$N+1%5" | 1     |
| [blank]\$N%[number]+[number] | " \$N%4+2" | 4     |
| <sup>a</sup> % is modulus.   |            |       |

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/NODExxxx, under the specified or default path on all database partitions. The *xxxx* is the database partition number as defined in the *db2nodes.cfg* file (that is, database partition 0 becomes NODE0000). Subdirectories SQL00001 through SQLnnnnn will reside on this path. This ensures that the database objects associated with different database partitions are stored in different directories (even if the subdirectory \$DB2INSTANCE under the specified or default path is shared by all database partitions).

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 the CATALOG LDAP DATABASE command.

CREATE DATABASE will fail if the application is already connected to a database.

When a database is created, a detailed deadlocks event monitor is created. As with any monitor, there is some overhead associated with this event monitor. You can drop the deadlocks event monitor by issuing the DROP EVENT MONITOR command.

Use CATALOG DATABASE to define different alias names for the new database.

### Examples:

Here are several examples of the CREATE DATABASE command:

#### Example 1:

```
CREATE DATABASE TESTDB3
AUTOMATIC STORAGE YES
```

Database TESTDB3 is created on the drive that is the value of database manager configuration parameter *dftdbpath*. Automatic storage is enabled with a single storage path that also has the value of *dftdbpath*.

#### Example 2:

```
CREATE DATABASE TESTDB7 ON C:.,D:
```



Database TESTDB7 is created on drive C: (first drive in storage path list). Automatic storage is implicitly enabled and the storage paths are C: and D:.

Example 3:

```
CREATE DATABASE TESTDB15
AUTOMATIC STORAGE YES
ON C: ,D: DBPATH ON E:
```

Database TESTDB15 is created on drive E: (explicitly listed as DBPATH). Automatic storage is explicitly enabled and the storage paths are C: and D:.

Example 4:

```
CREATE DATABASE TESTDB9 ON C:
USING CODESET UTF-8 TERRITORY US
```

Database TESTDB9 is created on drive C:. The codeset is set to UTF-8, enabling the use of native XML functionality on the database.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.

### Related concepts:

- “Isolation levels” in *SQL Reference, Volume 1*
- “Automatic storage databases” in *Administration Guide: Implementation*
- “Unicode implementation in DB2 Database for Linux, UNIX, and Windows” in *Administration Guide: Planning*

### Related tasks:

- “Creating a database” in *Administration Guide: Implementation*
- “Securing the system catalog view” in *Administration Guide: Implementation*
- “Collating Thai characters” in *Administration Guide: Planning*

### Related reference:

- “AUTOCONFIGURE ” on page 346
- “BIND” on page 355
- “CATALOG DATABASE ” on page 372
- “CATALOG LDAP DATABASE ” on page 377
- “DROP DATABASE ” on page 426
- “sqlcran API - Create a database on a database partition server” in *Administrative API Reference*
- “sqlcrea API - Create database” in *Administrative API Reference*
- “CREATE TABLESPACE statement” in *SQL Reference, Volume 2*
- “RESTORE DATABASE ” on page 675
- “auto\_maint - Automatic maintenance configuration parameter” in *Performance Guide*
- “Miscellaneous variables” in *Performance Guide*
- “self\_tuning\_mem- Self tuning memory configuration parameter” in *Performance Guide*

## CREATE TOOLS CATALOG

Creates the DB2 tools catalog tables in a new or existing database. The database must be local.

The tools catalog contains information about the administrative tasks that you configure with such tools as the Task Center and Control Center.

This command will optionally force all applications and stop and restart the database manager if new table spaces are created for the tools catalog. It will also update the DB2 Administration Server (DAS) configuration and activate the scheduler.

This command is not valid on a DB2 client.

### Scope:

The node from which this command is issued becomes the catalog node for the new database.

### Authorization:

One of the following:

- sysadm
- sysctrl

The user must also have DASADM authority to update the DB2 administration server configuration parameters.

### Required connection:

A database connection is temporarily established by this command during processing. This command will optionally stop and restart the database manager if new table spaces are created.

### Command syntax:

```

▶▶—CREATE TOOLS CATALOG—catalog-name—————▶
▶—CREATE NEW DATABASE—database-name—————▶
└—USE EXISTING—┬—TABLESPACE—tablespace-name—IN—┐ DATABASE—database-name—┘
▶—┬—FORCE—┐ ┬—KEEP INACTIVE—┐—————▶

```

### Command parameters:

#### CATALOG *catalog-name*

A name to be used to uniquely identify the DB2 tools catalog. The catalog tables are created under this schema name.

#### NEW 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.

**EXISTING DATABASE database-name**

The name of an existing database to host the tools catalog. It must be a local database.

**EXISTING TABLESPACE tablespace-name**

A name to be used to specify the existing 32K page table space used to create the DB2 tools catalog tables. A 32K page size temporary table space must also exist for the tables to be created successfully.

**FORCE**

When you create a tools catalog in a new table space, the database manager must be restarted, which requires that no applications be connected. Use the FORCE option to ensure that no applications are connected to the database. If applications are connected, the tools catalog creation will fail unless you specify an existing table space.

**KEEP INACTIVE**

This option will not update the DB2 administration server configuration parameters or enable the scheduler.

**Examples:**

```
db2 create tools catalog cc create new database toolsdb
```

```
db2 create tools catalog use existing database toolsdb force
```

```
db2 create tools catalog foobar use existing tablespace user32Ksp
in database toolsdb
```

```
db2 create tools catalog toolscat use existing database toolsdb keep inactive
```

**Usage notes:**

- The tools catalog tables require two 32K page table spaces (regular and temporary). In addition, unless you specify existing table spaces, a new 32K buffer pool is created for the table spaces. This requires a restart of the database manager. If the database manager must be restarted, all existing applications must be forced off. The new table spaces are created with a single container each in the default database directory path.
- If an active catalog with this name exists before you execute this command, it is deactivated and the new catalog becomes the active catalog.
- Multiple DB2 tools catalogs can be created in the same database and are uniquely identified by the catalog name.
- The *jdk\_path* configuration parameter must be set in the DB2 administration server (DAS) configuration to the minimum supported level of the SDK for Java.
- Updating the DAS configuration parameters requires *dasadm* authority on the DB2 administration server.
- Unless you specify the KEEP INACTIVE option, this command updates the local DAS configuration parameters related to the DB2 tools catalog database configuration and enables the scheduler at the local DAS server.
- The *jdk\_64\_path* configuration parameter must be set if you are creating a tools catalog against a 64-bit instance on one of the platforms that supports both 32- and 64-bit instances (AIX, HP-UX, and Solaris).

**Related concepts:**

- “DB2 Administration Server” in *Administration Guide: Implementation*

**Related reference:**

## CREATE TOOLS CATALOG

- “jdk\_64\_path - 64-Bit Software Developer's Kit for Java installation path DAS configuration parameter” in *Performance Guide*
- “jdk\_path - Software Developer's Kit for Java installation path DAS configuration parameter” in *Performance Guide*

## DEACTIVATE DATABASE

Stops the specified database.

### Scope:

In an MPP system, this command deactivates the specified database on all database partitions in the system. If one or more of these database partitions encounters an error, a warning is returned. The database will be successfully deactivated on some database partitions, but might continue to be active 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.)

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

### Related concepts:

- “Quick-start tips for performance tuning” in *Performance Guide*

## DEACTIVATE DATABASE

**Related reference:**

- “STOP DATABASE MANAGER ” on page 736
- “ACTIVATE DATABASE ” on page 330
- “sqlc\_deactivate\_db API - Deactivate database” in *Administrative API Reference*

---

## DECOMPOSE XML DOCUMENT

This command invokes a stored procedure to decompose a single XML document using a registered and decomposition-enabled XML schema..

**Authorization:**

One of the following groups of privileges or authorities is required:

- All of the following privileges:
  - INSERT privileges on the target table, as required for the operation specified in the action file
  - SELECT, INSERT, UPDATE or DELETE privileges as required, on any table referenced in the db2-xdb:expression or db2-xdb:condition annotation
- One of the following privileges or authorities:
  - CONTROL privilege on the target table
  - SYSADM or DBADM authority

**Required connection:**

Database.

**Command syntax:**

```

▶▶—DECOMPOSE XML DOCUMENT—xml-document-name—XMLSCHEMA—xml-schema-name————▶
▶
 [VALIDATE]

```

**Command parameters:**

**DECOMPOSE XML DOCUMENT** *xml-document-name*

*xml-document-name* is the file path and file name of the input XML document to be decomposed.

**XMLSCHEMA** *xml-schema-name*

*xml-schema-name* is the name of an existing XML schema registered with the XML schema repository to be used for document decomposition.

*xml-schema-name* is a qualified SQL identifier consisting of an optional SQL schema name followed by a period and the XML schema name. If the SQL schema name is not specified, it is assumed to be the value of the DB2 special register CURRENT SCHEMA.

**VALIDATE**

This parameter indicates that the input XML document is to be validated first, then decomposed only if the document is valid. If VALIDATE is not specified, the input XML document will not be validated before decomposition.

**Examples:**

The following example specifies that the XML document ~./gb/document1.xml is to be validated and decomposed with the registered XML schema DB2INST1.GENBANKSCHEMA.

```

DECOMPOSE XML DOCUMENT ./gb/document1.xml
 XMLSCHEMA DB2INST1.GENBANKSCHEMA
 VALIDATE

```

## DECOMPOSE XML DOCUMENT

The following example specifies that the XML document `./gb/document2.xml` is to be decomposed without validation with the registered XML schema `DB2INST2."GENBANK SCHEMA1"`, on the assumption that the value of the DB2 special register `CURRENT SCHEMA` is set to `DB2INST2`.

```
DECOMPOSE XML DOCUMENT ./gb/document2.xml
XMLSCHEMA "GENBANK SCHEMA1"
```

### Related concepts:

- “XML schema, DTD, and external entity management using the XML schema repository (XSR)” in *XML Guide*



## DEREGISTER

Deregisters the DB2 server from the network directory server.

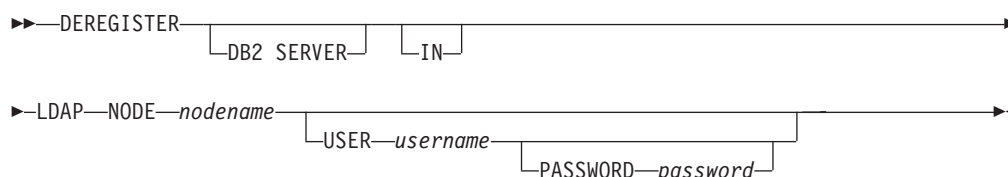
### Authorization:

None

### Required connection:

None

### Command syntax:



### Command parameters:

**IN** Specifies the network directory server from which to deregister the DB2 server. The valid value is LDAP for an LDAP (Lightweight Directory Access Protocol) directory server.

#### USER username

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 nodername

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.

### Related concepts:

- "Lightweight Directory Access Protocol (LDAP) overview" in *Administration Guide: Implementation*

### Related tasks:

- "Deregistering the DB2 server" in *Administration Guide: Implementation*

### Related reference:

## DEREGISTER

- “db2LdapDeregister API - Deregister the DB2 server and cataloged databases from the LDAP server” in *Administrative API Reference*
- “REGISTER ” on page 637
- “UPDATE LDAP NODE ” on page 780

## DESCRIBE

This command:

- Displays the output SQLDA information about a SELECT, CALL, or XQuery statement
- Displays columns of a table or a view
- Displays indexes of a table or a view
- Displays data partitions of a table or view

### Authorization:

To display the output 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, indexes or data partitions of a table or a view, SELECT privilege, CONTROL privilege, *sysadm* authority or *dbadm* authority is required for the following system catalogs:

- SYSCAT.COLUMNS (DESCRIBE TABLE), SYSCAT.DATAPARTITIONEXPRESSION (with SHOW DETAIL)
- SYSCAT.INDEXES (DESCRIBE INDEXES FOR TABLE) execute privilege on GET\_INDEX\_COLNAMES() UDF (with SHOW DETAIL)
- SYSCAT.DATAPARTITIONS (DESCRIBE DATA PARTITIONS FOR TABLE)

As PUBLIC has all the privileges over declared global temporary tables, a user can use the command to display information about any declared global temporary table that exists within its connection.

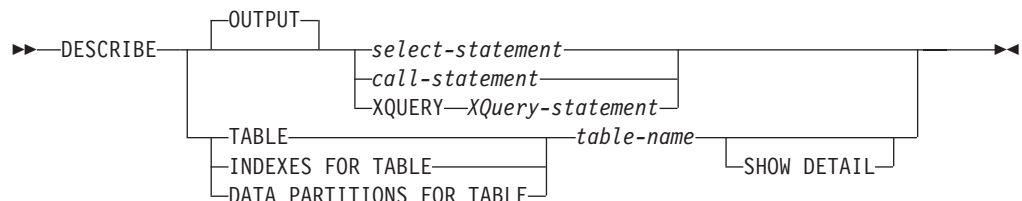
To display the output SQLDA information about a CALL statement, one of the privileges or authorities listed below is required:

- EXECUTE privilege on the stored procedure
- *sysadm* or *dbadm* authority

### Required connection:

Database. If implicit connect is enabled, a connection to the default database is established.

### Command syntax:



### Command parameters:

#### OUTPUT

Indicates that the output of the statement should be described. This keyword is optional.

## DESCRIBE

### **select-statement, call-statement, or XQUERY XQuery-statement**

Identifies the statement about which information is wanted. The statement is automatically prepared by CLP. To identify an XQuery statement, precede the statement with the keyword XQUERY.

### **TABLE table-name**

Specifies the table or view to be described. The fully qualified name in the form *schema.table-name* must be used. An alias for the table cannot be used in place of the actual table. 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 in the form *schema.table-name* must be used. An alias for the table cannot be used in place of the actual table. 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

### **DATA PARTITIONS FOR TABLE table-name**

Specifies the table or view for which data partitions need to be described. The information displayed for each data partition in the table includes; the partition identifier and the partitioning intervals. Results are ordered according to the partition identifier sequence. The fully qualified name in the form *schema.table-name* must be used. An alias for the table cannot be used in place of the actual table. The *schema* is the user name under which the table or view was created.

### **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
- Distribution key sequence
- Code page
- Default
- Table partitioning type (for tables partitioned by range this output appears below the original output)

- Partitioning key columns (for tables partitioned by range this output appears below the original output)

For the DESCRIBE INDEXES FOR TABLE command, specifies that output include the following additional information:

- Column names

For the DESCRIBE DATA PARTITIONS FOR TABLE command, specifies that output include a second table with the following additional information:

- Data partition sequence identifier
- Data partition expression in SQL

### Examples:

#### Describing the output a SELECT Statement

The following example shows how to describe a SELECT statement:

```
db2 "describe output 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 the output a CALL Statement

Given a stored procedure created with the statement:

```
CREATE PROCEDURE GIVE_BONUS (IN EMPNO INTEGER,
 IN DEPTNO INTEGER,
 OUT CHEQUE INTEGER,
 INOUT BONUS DEC(6,0))
...
```

The following example shows how to describe the output of a CALL statement:

## DESCRIBE

```
db2 "describe output call give_bonus(123456, 987, ?, 15000.)"
```

SQLDA Information

```
sqldaid :SQLDA sqldabc:896 sqln:20 sqld:2
```

Column Information

| sqltype     | sqllen | sqlname.data | sqlname.length |
|-------------|--------|--------------|----------------|
| -----       | -----  | -----        | -----          |
| 497 INTEGER | 4      |              |                |
| 485 DECIMAL | 6,0    |              |                |

### Describing the output of an XQuery Statement

Given a table named CUSTOMER that has a column named INFO of the XML data type, the following example shows how to describe an XQuery statement:

```
db2 'describe xquery for $cust in db2-fn:xmlcolumn("CUSTOMER.INFO") return $cust'
```

SQLDA Information

```
sqldaid : SQLDA sqldabc: 1136 sqln: 20 sqld: 1
```

Column Information

| sqltype | sqllen | sqlname.data | sqlname.length |
|---------|--------|--------------|----------------|
| -----   | -----  | -----        | -----          |
| 998 XML | 0 1    |              | 1              |

If the keyword XQUERY is not specified, SQL0104N is returned.

```
db2 'describe for $cust in db2-fn:xmlcolumn("CUSTOMER.INFO") return $cust'
SQL0104N An unexpected token "for" was found following "DESCRIBE". Expected
tokens may include: "OUTPUT". SQLSTATE=42601
```

If the DESCRIBE XQUERY command is issued against a downlevel server that does not support the XQUERY option, the message DB21108E is returned to indicate that the functionality is not supported by the downlevel server.

### Describing a Table

The following example shows how to describe a table:

```
db2 describe table user1.department
```

Table: USER1.DEPARTMENT

| Column name | Type schema | Type name | Length | Scale | Nulls |
|-------------|-------------|-----------|--------|-------|-------|
| -----       | -----       | -----     | -----  | ----- | ----- |
| AREA        | SYSIBM      | SMALLINT  |        | 2     | 0 No  |
| DEPT        | SYSIBM      | CHARACTER |        | 3     | 0 No  |
| DEPTNAME    | SYSIBM      | CHARACTER |        | 20    | 0 Yes |

The following example shows how to describe a table with details. If the table is partitioned, as in this example, additional details appear below the existing output. For a non-partitioned table, the additional table heading is not displayed:

```
db2 describe table user1.employee show detail
```

| Column name | Type schema | Column number | Type name | Length |
|-------------|-------------|---------------|-----------|--------|
| FIRST       | SYSIBM      | 0             | CHARACTER | 10     |
| LAST        | SYSIBM      | 1             | CHARACTER | 10     |

Table is partitioned by range (ordered on the following column/s):

-----  
LAST  
FIRST

### 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 schema | Index name | Unique rule | Number of columns |
|--------------|------------|-------------|-------------------|
| USER1        | IDX1       | U           | 2                 |

### Describing Data Partitions

The following example shows how to describe data partitions:

```
db2 describe data partitions for table user1.sales
```

| PartitionId | Inclusive (y/n)<br>Low Value | Inclusive (y/n)<br>High Value |
|-------------|------------------------------|-------------------------------|
| 0           | Y 2001,1                     | Y 2001,3                      |
| 1           | N 2001,3                     | Y 2001,6                      |
| 3           | N 2001,6                     | Y 2001,9                      |

Describing the data partitions with details returns the same output as in the previous example including an additional table showing the Partition Id and table space where the data for the data partition is stored:

```
db2 describe data partitions for table user1.employee show detail
```

| PartitionId | Inclusive (y/n)<br>Low Value | Inclusive (y/n)<br>High Value |
|-------------|------------------------------|-------------------------------|
| 0           | Y MINVALUE,MINVALUE          | Y 'beck','kevin'              |
| 1           | N 'beck','kevin'             | N 'treece','jeff'             |
| 2           | Y 'treece','jeff'            | Y 'zhang','liping'            |
| 3           | Y 'zyzyck',MINVALUE          | Y MAXVALUE,MAXVALUE           |

| PartitionId | PartitionName | TableSpId | LongTblSpId | ObjectId | AccessMode | Status |
|-------------|---------------|-----------|-------------|----------|------------|--------|
| 0           | PARTx         | 3         | 43          | F        |            |        |
| 1           | PARTNew       | 13        | 13          | N        | A          |        |
| 2           | PART3         | 31        | 33          | F        |            |        |
| 3           | PART4         | 23        | 34          | N        | A          |        |

## DESCRIBE

### Related reference:

- “ADMIN\_CMD procedure – Run administrative commands” in *Administrative SQL Routines and Views*
- “DESCRIBE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “SYSCAT.DATAPARTITIONS catalog view” in *SQL Reference, Volume 1*



---

## 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

**Related tasks:**

- “Attaching to and detaching from a non-default instance of the database manager” in *Administration Guide: Implementation*

**Related reference:**

- “ATTACH ” on page 344
- “sqledtin API - Detach from instance” in *Administrative API Reference*

---

## DROP CONTACT

Removes a contact from the list of contacts defined on the local system. A contact is a user to whom the Scheduler and Health Monitor send messages. The setting of the Database Administration Server (DAS) *contact\_host* configuration parameter determines whether the list is local or global.

**Authorization:**

None.

**Required connection:**

None. Local execution only: this command cannot be used with a remote connection.

**Command syntax:**

►►—DROP CONTACT—*name*—————►◄

**Command parameters:**

**CONTACT name**

The name of the contact that will be dropped from the local system.

**Related reference:**

- “db2DropContact API - Remove a contact from the list of contacts to whom notification messages can be sent” in *Administrative API Reference*
- “DROP CONTACT command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

---

## DROP CONTACTGROUP

Removes a contact group from the list of contacts defined on the local system. A contact group contains a list of users to whom the Scheduler and Health Monitor send messages. The setting of the Database Administration Server (DAS) *contact\_host* configuration parameter determines whether the list is local or global.

**Authorization:**

None.

**Required Connection:**

None.

**Command Syntax:**

►►—DROP CONTACTGROUP—*name*—————►◄

**Command Parameters:****CONTACTGROUP name**

The name of the contact group that will be dropped from the local system.

**Related reference:**

- “db2DropContactGroup API - Remove a contact group from the list of contacts to whom notification messages can be sent” in *Administrative API Reference*
- “DROP CONTACTGROUP command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

## 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 database partitions 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:



### 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 DBPARTITIONNUM

Specifies that the database is to be deleted only on the database partition that issued the DROP DATABASE command. This parameter is used by utilities supplied with DB2 ESE, 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, as well as any back/restore history for the database. If the log files are needed for a roll-forward recovery after a restore operation, or the backup history required to restore the database, these 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 might not be seen immediately on the DB2 Data Links Manager, and the unlinked files might 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.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.

### Related tasks:

- "Dropping a database" in *Administration Guide: Implementation*

### Related reference:

- "CREATE DATABASE " on page 395
- "UNCATALOG DATABASE " on page 745
- "sqledpan API - Drop a database on a database partition server" in *Administrative API Reference*
- "sqledrpd API - Drop database" in *Administrative API Reference*
- "CATALOG DATABASE " on page 372

---

# DROP DBPARTITIONNUM VERIFY

Verifies if a database partition exists in the database partition groups of any databases, and if an event monitor is defined on the database partition. This command should be used prior to dropping a database partition from a partitioned database system.

### Scope:

This command only affects the database partition on which it is issued.

### Authorization:

*sysadm*

### Command syntax:

►►—DROP DBPARTITIONNUM VERIFY—◀◀

### Command parameters:

None

### Usage notes:

If a message is returned, indicating that the database partition is not in use, use the STOP DATABASE MANAGER command with DROP DBPARTITIONNUM to remove the entry for the database partition from the `db2nodes.cfg` file, which removes the database partition from the database system.

If a message is returned, indicating that the database partition is in use, the following actions should be taken:

1. If the database partition contains data, redistribute the data to remove it from the database partition using REDISTRIBUTE DATABASE PARTITION GROUP. Use either the DROP DBPARTITIONNUM option on the REDISTRIBUTE DATABASE PARTITION GROUP command or on the ALTER DATABASE PARTITION GROUP statement to remove the database partition from any database partition groups for the database. This must be done for each database that contains the database partition in a database partition group.
2. Drop any event monitors that are defined on the database partition.
3. Rerun DROP DBPARTITIONNUM VERIFY to ensure that the database is no longer in use.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.

### Related reference:

- “STOP DATABASE MANAGER ” on page 736
- “REDISTRIBUTE DATABASE PARTITION GROUP ” on page 633
- “sqlldrpn API - Check whether a database partition server can be dropped” in *Administrative API Reference*

## DROP TOOLS CATALOG

Drops the DB2 tools catalog tables for the specified catalog in the given database. This command is not valid on a DB2 client.

**Warning:** If you drop the active tools catalog, you can no longer schedule tasks and scheduled tasks are not executed. To activate the scheduler, you must activate a previous tools catalog or create a new one.

### Scope:

This command affects the database.

### Authorization:

One of the following:

- sysadm
- sysctrl

The user must also have DASADM authority to update the DB2 administration server (DAS) configuration parameters.

### Required connection:

A database connection is temporarily established by this command during processing.

### Command syntax:

```
►►—DROP TOOLS CATALOG—catalog-name—IN DATABASE—database-name—┐FORCE┘—►►
```

### Command parameters:

#### CATALOG *catalog-name*

A name to be used to uniquely identify the DB2 tools catalog. The catalog tables are dropped from this schema.

#### DATABASE *database-name*

A name to be used to connect to the local database containing the catalog tables.

#### FORCE

The force option is used to force the DB2 administration server's scheduler to stop. If this is not specified, the tools catalog will not be dropped if the scheduler cannot be stopped.

### Examples:

```
db2 drop tools catalog cc in database toolsdb
db2 drop tools catalog in database toolsdb force
```

### Usage notes:

- The *jdk\_path* configuration parameter must be set in the DB2 administration server (DAS) configuration to the minimum supported level of the SDK for Java.
- This command will disable the scheduler at the local DAS and reset the DAS configuration parameters related to the DB2 tools catalog database configuration.

## DROP TOOLS CATALOG

### Related tasks:

- “Creating a database for the DB2 tools catalog” in *Administration Guide: Implementation*



---

## 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.

---

**EDIT**

Launches a user-specified editor with a specified command for editing. When the user finishes editing, saves the contents of the editor and exits the editor, permits the user to execute the command in CLP interactive mode.

**Scope**

This command can only be run within CLP interactive mode. Specifically, it cannot be run from the CLP command mode or the CLP batch mode.

**Authorization:**

None

**Required connection:**

None

**Command syntax:**

```

▶▶ [EDIT] [E] [EDITOR editor] [num]

```

**Command parameters:****EDITOR**

Launch the editor specified for editing. If this parameter is not specified, the editor to be used is determined in the following order:

1. the editor specified by the DB2\_CLP\_EDITOR registry variable
2. the editor specified by the VISUAL environment variable
3. the editor specified by the EDITOR environment variable
4. On Windows operating systems, the Notepad editor; on UNIX operating systems, the vi editor

**num** If *num* is positive, launches the editor with the command corresponding to *num*. If *num* is negative, launches the editor with the command corresponding to *num*, counting backwards from the most recent command in the command history. Zero is not a valid value for *num*. If this parameter is not specified, launches the editor with the most recently run command. (This is equivalent to specifying a value of -1 for *num*.)

**Usage notes:**

1. The editor specified must be a valid editor contained in the PATH of the operating system.
2. You can view a list of the most recently run commands available for editing by executing the HISTORY command.
3. The EDIT command will never be recorded in the command history. However, if you choose to run a command that was edited using the EDIT command, this command will be recorded in the command history.

**Related reference:**

- "HISTORY " on page 493

## 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:

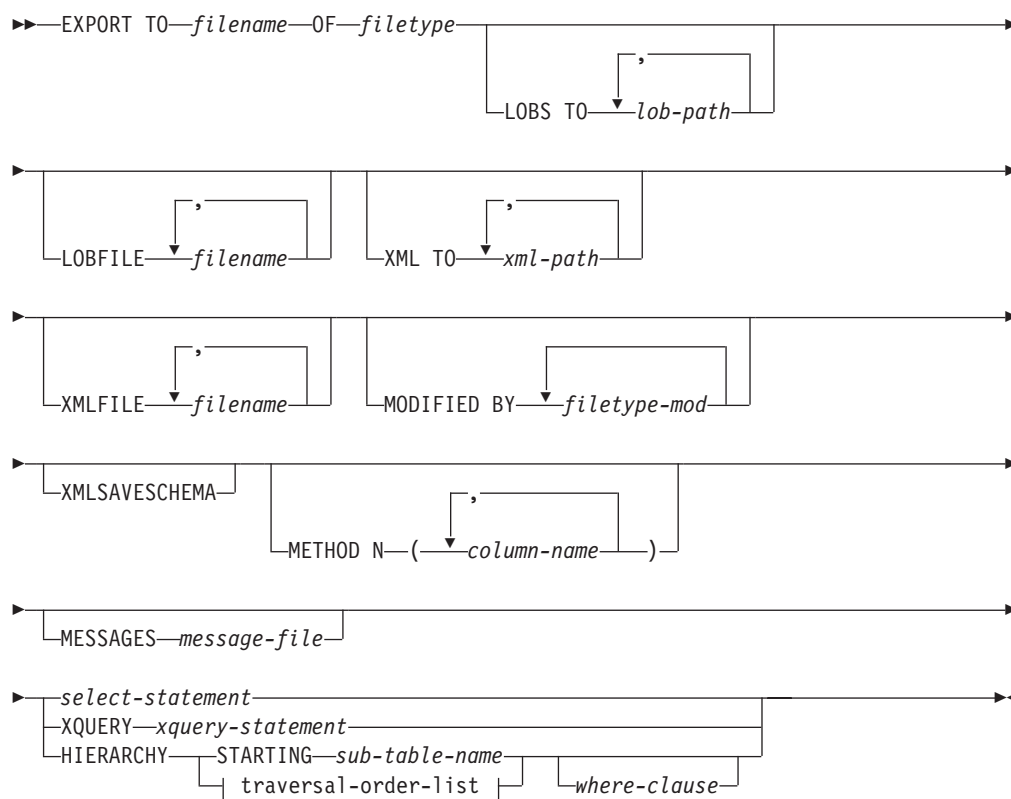
- *sysadm*
- *dbadm*

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. Utility access to Linux, UNIX, or Windows database servers from Linux, UNIX, or Windows clients must be a direct connection through the engine and not through a DB2 Connect™ gateway or loop back environment.

### 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. The maximum number of file names that can be specified is 999. This will implicitly activate the LOBSINFILE behavior.

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 and the three character identifier *lob*. 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.lob*, */u/foo/lob/path/bar.002.lob*, and so on.

**LOBS TO lob-path**

Specifies one or more paths to directories in which the LOB files are to be stored. There will be at least one file per LOB path, and each file will contain at least one LOB. The maximum number of paths that can be specified is 999. This will implicitly activate the LOBSINFILE behavior.

**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 file type modifier options. See File type modifiers for the export utility.

**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

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 might 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.

#### **select-statement**

Specifies the SELECT or XQUERY statement that will return the data to be exported. If the 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.

#### **XMLFILE filename**

Specifies one or more base file names for the XML files. When name space is exhausted for the first name, the second name is used, and so on.

When creating XML files during an export operation, file names are constructed by appending the current base name from this list to the current path (from xml-path), appending a 3-digit sequence number, and appending the three character identifier xml. For example, if the current XML path is the directory /u/foo/xml/path/, and the current XML file name is bar, the XML files created will be /u/foo/xml/path/bar.001.xml, /u/foo/xml/path/bar.002.xml, and so on.

#### **XML TO xml-path**

Specifies one or more paths to directories in which the XML files are to be stored. There will be at least one file per XML path, and each file will contain at least one XQuery Data Model (QDM) instance. If more than one path is specified, then QDM instances are distributed evenly among the paths.

#### **XMLSAVESCHEMA**

Specifies that XML schema information should be saved for all XML columns. For each exported XML document that was validated against an XML schema when it was inserted, the fully qualified SQL identifier of that schema will be stored as an (SCH) attribute inside the corresponding XML Data Specifier (XDS). If the exported document was not validated against an XML schema or the schema object no longer exists in the database, an SCH attribute will not be included in the corresponding XDS.

The schema and name portions of the SQL identifier are stored as the "OBJECTSCHEMA" and "OBJECTNAME" values in the row of the SYSCAT.XSROBJECTS catalog table corresponding to the XML schema.

The XMLSAVESCHEMA option is not compatible with XQuery sequences that do not produce well-formed XML documents.

### Examples:

The following example shows how to export information from the STAFF table in the SAMPLE database to the file myfile.ixf. The output will be in IXF format. You must be connected to the SAMPLE database before issuing the command. The index definitions (if any) will be stored in the output file except when the database connection is made through DB2 Connect.

```
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. The output will be in IXF format and will go into the awards.ixf file. You must first connect to the SAMPLE database before issuing the command. Also, the actual column name in the table is 'dept' instead of 'department'.

```
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 a 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 might 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 charde1' 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, fields containing the row separators will shrink or expand.
- 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<sup>®</sup> servers such as DB2 for OS/390<sup>®</sup>, DB2 for VM and VSE, and DB2 for OS/400<sup>®</sup>. 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.
- Use the same traverse order during an import operation. The load utility does not support loading hierarchies or sub-hierarchies.
- When exporting data from a table that has protected rows, the LBAC credentials held by the session authorization id might limit the rows that are exported. Rows that the session authorization ID does not have read access to will not be exported. No error or warning is given.
- If the LBAC credentials held by the session authorization id do not allow reading from one or more protected columns included in the export then the export fails and an error (SQLSTATE 42512) is returned.
- Export packages are bound using DATETIME ISO format, thus, all date/time/timestamp values are converted into ISO format when cast to a string representation. Since the CLP packages are bound using DATETIME LOC format (locale specific format), you may see inconsistent behaviour between CLP and export if the CLP DATETIME format is different from ISO. For instance, the following SELECT statement may return expected results:

```
db2 select col2 from tab1 where char(col2)='05/10/2005';
COL2

05/10/2005
05/10/2005
05/10/2005
3 record(s) selected.
```

But an export command using the same select clause will not:

```
db2 export to test.del of del select col2 from test
where char(col2)='05/10/2005';
Number of rows exported: 0
```

Now, replacing the LOCALE date format with ISO format gives the expected results:

```
db2 export to test.del of del select col2 from test
where char(col2)='2005-05-10';
Number of rows exported: 3
```

#### Related concepts:

- “Export Overview” in *Data Movement Utilities Guide and Reference*
- “Privileges, authorities and authorization required to use export” in *Data Movement Utilities Guide and Reference*

## EXPORT

### Related tasks:

- “Exporting data” in *Data Movement Utilities Guide and Reference*

### Related reference:

- “ADMIN\_CMD procedure – Run administrative commands” in *Administrative SQL Routines and Views*
- “EXPORT command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “Export Sessions - CLP Examples” in *Data Movement Utilities Guide and Reference*
- “LOB and XML file behavior with regard to import and export” in *Data Movement Utilities Guide and Reference*



## 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 database partitions that are listed in the \$HOME/sql11ib/db2nodes.cfg file.

In a partitioned database environment, this command does not have to be issued from the coordinator database partition of the application being forced. It can be issued from any node (database partition server) in the partitioned database environment.

### Authorization:

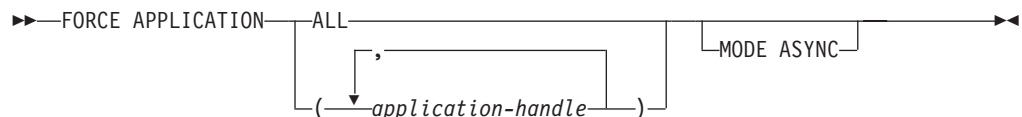
One of the following:

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

### 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:



### Command parameters:

#### APPLICATION

**ALL** All applications will be disconnected from the database.

#### application-handle

Specifies the agent to be terminated. List the values using the LIST APPLICATIONS command.

#### 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:

## FORCE APPLICATION

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:

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 might be required to completely force all users off.

### Related reference:

- "LIST APPLICATIONS " on page 520
- "ATTACH " on page 344
- "sqlfrce API - Force users and applications off the system" in *Administrative API Reference*
- "FORCE APPLICATION command using the ADMIN\_CMD procedure" in *Administrative SQL Routines and Views*
- "APPLICATIONS administrative view – Retrieve connected database application information" in *Administrative SQL Routines and Views*

## GET ADMIN CONFIGURATION

Returns the values of individual DB2 Administration Server (DAS) configuration parameter values on the administration node of the system. The DAS is a special administrative tool that enables remote administration of DB2 servers. For a list of the DAS configuration parameters, see the description of the UPDATE ADMIN CONFIGURATION command.

### Scope:

This command returns information about DAS configuration parameters on the administration node of the system to which you are attached or that you specify in the FOR NODE option.

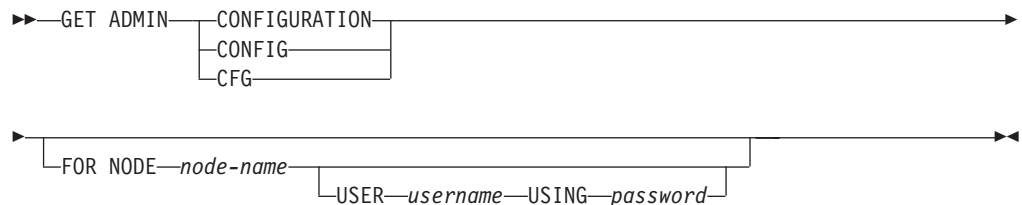
### Authorization:

None

### Required connection:

Node. To display the DAS configuration for a remote system, first connect to that system or use the FOR NODE option to specify the administration node of the system.

### Command syntax:



### Command parameters:

#### FOR NODE

Enter the name of the administration node to view DAS configuration parameters there.

#### USER *username* USING *password*

If connection to the node requires user name and password, enter this information.

### Examples:

The following is sample output from GET ADMIN CONFIGURATION:

## GET ADMIN CONFIGURATION

### Admin Server Configuration

|                                            |                                   |
|--------------------------------------------|-----------------------------------|
| Authentication Type DAS                    | (AUTHENTICATION) = SERVER_ENCRYPT |
| DAS Administration Authority Group Name    | (DASADM_GROUP) = ADMINISTRATORS   |
| DAS Discovery Mode                         | (DISCOVER) = SEARCH               |
| Name of the DB2 Server System              | (DB2SYSTEM) = swalkty             |
| Java Development Kit Installation Path DAS | (JDK_PATH) = e:\sql1lib\java\jdk  |
| DAS Code Page                              | (DAS_CODEPAGE) = 0                |
| DAS Territory                              | (DAS_TERRITORY) = 0               |
| Location of Contact List                   | (CONTACT_HOST) = hostA.ibm.ca     |
| Execute Expired Tasks                      | (EXEC_EXP_TASK) = NO              |
| Scheduler Mode                             | (SCHED_ENABLE) = ON               |
| SMTP Server                                | (SMTP_SERVER) = smtp1.ibm.ca      |
| Tools Catalog Database                     | (TOOLSCAT_DB) = CCMD              |
| Tools Catalog Database Instance            | (TOOLSCAT_INST) = DB2             |
| Tools Catalog Database Schema              | (TOOLSCAT_SCHEMA) = TOOLSCAT      |
| Scheduler User ID                          | = db2admin                        |

#### Usage notes:

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 DAS again to recover.

To set the configuration parameters to the default values shipped with the DAS, use the RESET ADMIN CONFIGURATION command.

#### Related reference:

- "RESET ADMIN CONFIGURATION " on page 663
- "UPDATE ADMIN CONFIGURATION " on page 756
- "Configuration parameters summary" in *Performance Guide*

## GET ALERT CONFIGURATION

Returns the alert configuration settings for health indicators for a particular instance.

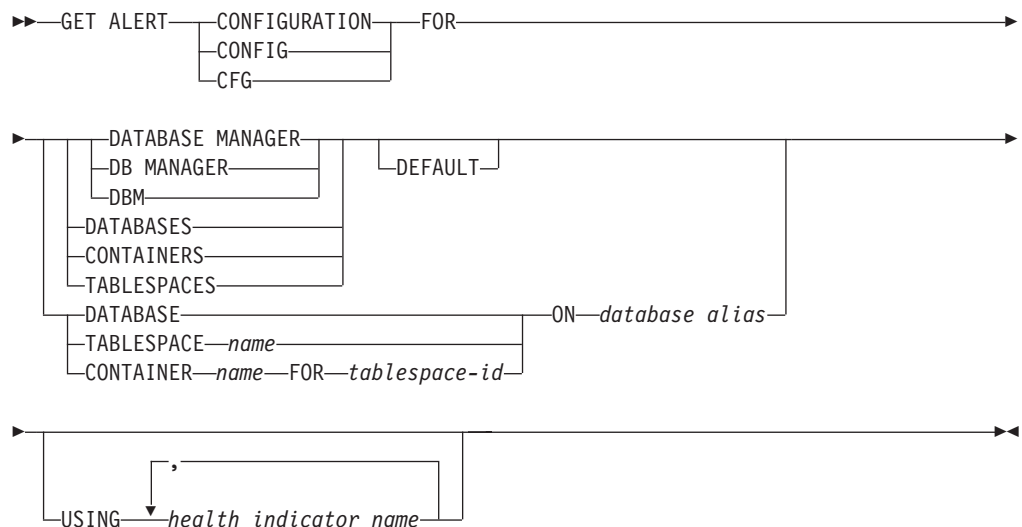
### Authorization:

None.

### Required connection:

Instance. An explicit attachment is not required.

### Command syntax:



### Command parameters:

#### DATABASE MANAGER

Retrieves alert settings for the database manager.

#### DATABASES

Retrieves alert settings for all databases managed by the database manager. These are the settings that apply to all databases that do not have custom settings. Custom settings are defined using the `DATABASE ON database alias` clause.

#### CONTAINERS

Retrieves alert settings for all table space containers managed by the database manager. These are the settings that apply to all table space containers that do not have custom settings. Custom settings are defined using the `"CONTAINER name ON database alias"` clause.

#### TABLESPACES

Retrieves alert settings for all table spaces managed by the database manager. These are the settings that apply to all table spaces that do not have custom settings. Custom settings are defined using the `TABLESPACE name ON database alias` clause.

#### DEFAULT

Specifies that the install defaults are to be retrieved.

## GET ALERT CONFIGURATION

### DATABASE ON *database alias*

Retrieves the alert settings for the database specified using the ON *database alias* clause. If this database does not have custom settings, then the settings for all databases for the instance will be returned, which is equivalent to using the DATABASES parameter.

### CONTAINER *name* FOR *tablespace-id* ON *database alias*

Retrieves the alert settings for the table space container called *name*, for the table space specified using the "FOR *tablespace-id*" clause, on the database specified using the "ON *database alias*" clause. If this table space container does not have custom settings, then the settings for all table space containers for the database will be returned, which is equivalent to using the CONTAINERS parameter.

### TABLESPACE *name* ON *database alias*

Retrieves the alert settings for the table space called *name*, on the database specified using the ON *database alias* clause. If this table space does not have custom settings, then the settings for all table spaces for the database will be returned, which is equivalent to using the TABLESPACES parameter.

### USING *health indicator name*

Specifies the set of health indicators for which alert configuration information will be returned. Health indicator names consist of a two-letter object identifier followed by a name that describes what the indicator measures. For example: db.sort\_privmem\_util. This is an optional clause, meaning that if it is not used, all health indicators for the specified object or object type will be returned.

### Examples:

The following is typical output resulting from a request for database manager information:

```
DB2 GET ALERT CFG FOR DBM
```

```
Alert Configuration
Indicator Name = db2.db2_op_status
Default = Yes
Type = State-based
Sensitivity = 0
Formula = db2.db2_status;
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db2.sort_privmem_util
Default = Yes
Type = Threshold-based
Warning = 90
Alarm = 100
Unit = %
Sensitivity = 0
Formula = ((db2.sort_heap_allocated/sheapthres)
 *100);
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db2.mon_heap_util
Default = Yes
Type = Threshold-based
Warning = 85
Alarm = 95
Unit = %
```

## GET ALERT CONFIGURATION

```
Sensitivity = 0
Formula = ((db2.mon_heap_cur_size/
 db2.mon_heap_max_size)*100);
Actions = Disabled
Threshold or State checking = Enabled
```

The following is typical output resulting from a request for configuration information:

DB2 GET ALERT CFG FOR DATABASES

```
Alert Configuration
Indicator Name = db.db_op_status
Default = Yes
Type = State-based
Sensitivity = 0
Formula = db.db_status;
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.sort_shrmem_util
Default = Yes
Type = Threshold-based
Warning = 70
Alarm = 85
Unit = %
Sensitivity = 0
Formula = ((db.sort_shrheap_allocated/sheaphres_shr)
 *100);
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.spilled_sorts
Default = Yes
Type = Threshold-based
Warning = 30
Alarm = 50
Unit = %
Sensitivity = 0
Formula = ((delta(db.sort_overflows,10))/
 (delta(db.total_sorts,10)+1)*100);
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.max_sort_shrmem_util
Default = Yes
Type = Threshold-based
Warning = 60
Alarm = 30
Unit = %
Sensitivity = 0
Formula = ((db.max_shr_sort_mem/
 sheaphres_shr)*100);
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.log_util
Default = Yes
Type = Threshold-based
Warning = 75
Alarm = 85
Unit = %
Sensitivity = 0
Formula = (db.total_log_used/
 (db.total_log_used+db.total_log_available)
)*100;
Actions = Disabled
```

## GET ALERT CONFIGURATION

```
Threshold or State checking = Enabled

Indicator Name = db.log_fs_util
Default = Yes
Type = Threshold-based
Warning = 75
Alarm = 85
Unit = %
Sensitivity = 0
Formula = ((os.fs_used/os.fs_total)*100);
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.deadlock_rate
Default = Yes
Type = Threshold-based
Warning = 5
Alarm = 10
Unit = Deadlocks per hour
Sensitivity = 0
Formula = delta(db.deadlocks);
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.locklist_util
Default = Yes
Type = Threshold-based
Warning = 75
Alarm = 85
Unit = %
Sensitivity = 0
Formula = (db.lock_list_in_use/(locklist*4096))
 *100;
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.lock_escal_rate
Default = Yes
Type = Threshold-based
Warning = 5
Alarm = 10
Unit = Lock escalations per hour
Sensitivity = 0
Formula = delta(db.lock_escal);
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.apps_waiting_locks
Default = Yes
Type = Threshold-based
Warning = 50
Alarm = 70
Unit = %
Sensitivity = 0
Formula = (db.locks_waiting/db.appls_cur_cons)*100;
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.pkgcache_hitratio
Default = Yes
Type = Threshold-based
Warning = 80
Alarm = 70
Unit = %
Sensitivity = 0
Formula = (1-
```



## GET ALERT CONFIGURATION

```

 (db.pkg_cache_inserts/db.pkg_cache_lookups)
)*100;
 Actions = Disabled
 Threshold or State checking = Disabled

Indicator Name = db.catcache_hitratio
Default = Yes
Type = Threshold-based
Warning = 80
Alarm = 70
Unit = %
Sensitivity = 0
Formula = (1-
 (db.cat_cache_inserts/db.cat_cache_lookups)
)*100;

 Actions = Disabled
 Threshold or State checking = Disabled

Indicator Name = db.shrworkspace_hitratio
Default = Yes
Type = Threshold-based
Warning = 80
Alarm = 70
Unit = %
Sensitivity = 0
Formula = ((1-
 (db.shr_workspace_section_inserts/
 db.shr_workspace_section_lookups))
 *100);

 Actions = Disabled
 Threshold or State checking = Disabled

Indicator Name = db.db_heap_util
Default = Yes
Type = Threshold-based
Warning = 85
Alarm = 95
Unit = %
Sensitivity = 0
Formula = ((db.db_heap_cur_size/
 db.db_heap_max_size)*100);

 Actions = Disabled
 Threshold or State checking = Enabled

Indicator Name = db.tb_reorg_req
Default = Yes
Type = Collection state-based
Sensitivity = 0
Actions = Disabled
Threshold or State checking = Disabled

Indicator Name = db.hadr_op_status
Default = Yes
Type = State-based
Sensitivity = 0
Formula = db.hadr_connect_status;
Actions = Disabled
Threshold or State checking = Enabled

Indicator Name = db.hadr_delay
Default = Yes
Type = Threshold-based
Warning = 10
Alarm = 15
Unit = Minutes
Sensitivity = 0
Formula = (db.hadr_log_gap*var.refresh_rate/60)

```

## GET ALERT CONFIGURATION

```

 DIV(delta(db.hadr_secondary_log_pos));
 Actions = Disabled
 Threshold or State checking = Enabled

Indicator Name = db.db_backup_req
 Default = Yes
 Type = State-based
 Sensitivity = 0
 Actions = Disabled
 Threshold or State checking = Disabled

Indicator Name = db.fed_nicknames_op_status
 Default = Yes
 Type = Collection state-based
 Sensitivity = 0
 Actions = Disabled
 Threshold or State checking = Disabled

Indicator Name = db.fed_servers_op_status
 Default = Yes
 Type = Collection state-based
 Sensitivity = 0
 Actions = Disabled
 Threshold or State checking = Disabled

Indicator Name = db.tb_runstats_req
 Default = Yes
 Type = Collection state-based
 Sensitivity = 0
 Actions = Disabled
 Threshold or State checking = Disabled
```

### Related tasks:

- “Configuring health indicators using a client application” in *System Monitor Guide and Reference*

### Related reference:

- “db2GetAlertCfg API - Get the alert configuration settings for the health indicators” in *Administrative API Reference*
- “HEALTH\_GET\_ALERT\_ACTION\_CFG table function –Retrieve health alert action configuration settings” in *Administrative SQL Routines and Views*
- “HEALTH\_GET\_ALERT\_CFG table function – Retrieve health alert configuration settings” in *Administrative SQL Routines and Views*

---

## 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 SYSMANT 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
Direct LOAD authority = YES
Direct QUIESCE_CONNECT authority = YES
Direct CREATE_EXTERNAL_ROUTINE = YES

Indirect SYSADM authority = YES
Indirect SYSCTRL authority = NO
Indirect SYSMANT 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
Indirect LOAD authority = NO
Indirect QUIESCE_CONNECT authority = NO
Indirect CREATE_EXTERNAL_ROUTINE = NO
```

**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. (All users belong to a special group called PUBLIC)
- The **GET AUTHORIZATIONS** command does not display whether or not the current user holds SECADM authority. To find out who holds SECADM authority, use the following query:

## GET AUTHORIZATIONS

```
SELECT GRANTEE FROM SYSCAT.DBAUTH
WHERE SECURITYADMAUTH = 'Y'
```

### **Related concepts:**

- “Authorization” in *Administration Guide: Planning*

### **Related reference:**

- “SYSCAT.DBAUTH catalog view” in *SQL Reference, Volume 1*

## 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.

### Authorization:

None

### Required connection:

None

### Command syntax:



### Command parameters:

#### AT GLOBAL LEVEL

Displays the default CLI configuration parameters in the LDAP directory. This parameter is only valid on Windows operating systems.

#### 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:

```

[tstcli1x]
uid=userid
pwd=password
autocommit=0
TableType="'TABLE','VIEW','SYSTEM TABLE'"

[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.

## GET CLI CONFIGURATION

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.

In an LDAP environment, users can configure a set of default CLI settings for a database catalogued in the LDAP directory. When an LDAP catalogued database is added as a Data Source Name (DSN), either by using the Configuration Assistant (CA) or the CLI/ODBC configuration utility, any default CLI settings, if they exist in the LDAP directory, will be configured for that DSN on the local machine. The AT GLOBAL LEVEL clause must be specified to display the default CLI settings.

### **Related reference:**

- "UPDATE CLI CONFIGURATION " on page 766
- "REFRESH LDAP " on page 636

---

## 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).

This command also returns information about:

- the database connection mode (SHARE or EXCLUSIVE)
- the alias and name of the database to which a connection exists (if one exists)
- the host name and service name of the connection if the connection is using TCP/IP

### 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
Hostname = montero
Service name = 29384

```

### Usage notes:

This command does not apply to type 2 connections.

### Related reference:

- "SET CLIENT " on page 716
- "UPDATE ALTERNATE SERVER FOR DATABASE" on page 762

---

# GET CONTACTGROUP

Returns the contacts included in a single contact group that is defined on the local system. A contact is a user to whom the Scheduler and Health Monitor send messages. You create named groups of contacts with the ADD CONTACTGROUP command.

**Authorization:**

None.

**Required connection:**

None. Local execution only: this command cannot be used with a remote connection.

**Command syntax:**

►► GET CONTACTGROUP *name* ◀◀

**Command parameters:**

**CONTACTGROUP name**

The name of the group for which you would like to retrieve the contacts.

**Examples:**

GET CONTACTGROUP support

Description

-----

Foo Widgets broadloom support unit

| Name    | Type          |
|---------|---------------|
| -----   | -----         |
| joe     | contact       |
| support | contact group |
| joline  | contact       |

**Related reference:**

- “db2GetContactGroup API - Get the list of contacts in a single contact group to whom notification messages can be sent” in *Administrative API Reference*
- “CONTACTGROUPS administrative view – Retrieve the list of contact groups” in *Administrative SQL Routines and Views*



---

## GET CONTACTGROUPS

The command provides a list of contact groups, which can be either defined locally on the system or in a global list. A contact group is a list of addresses to which monitoring processes such as the Scheduler and Health Monitor can send messages. The setting of the Database Administration Server (DAS) *contact\_host* configuration parameter determines whether the list is local or global. You create named groups of contacts with the ADD CONTACTGROUP command.

**Authorization:**

None

**Required Connection:**

None

**Command Syntax:**

▶▶—GET CONTACTGROUPS—◀◀

**Command Parameters:**

None

**Examples:**

In the following example, the command GET CONTACTGROUPS is issued. The result is as follows:

| Name    | Description                          |
|---------|--------------------------------------|
| -----   | -----                                |
| support | Foo Widgets broadloom support unit   |
| service | Foo Widgets service and support unit |

**Related reference:**

- “db2GetContactGroups API - Get the list of contact groups to whom notification messages can be sent” in *Administrative API Reference*
- “CONTACTGROUPS administrative view – Retrieve the list of contact groups” in *Administrative SQL Routines and Views*

---

## GET CONTACTS

Returns the list of contacts defined on the local system. Contacts are users to whom the monitoring processes such as the Scheduler and Health Monitor send notifications or messages.

To create a contact, use the ADD CONTACT command.

**Authorization:**

None.

**Required connection:**

None.

**Command syntax:**

▶▶—GET CONTACTS—◀◀

**Examples:**

GET CONTACTS

| Name   | Type   | Address                      | Max Page Length | Description  |
|--------|--------|------------------------------|-----------------|--------------|
| joe    | e-mail | joe@somewhere.com            | -               | -            |
| joline | e-mail | joline@<br>somewhereelse.com | -               | -            |
| john   | page   | john@relay.org               | 50              | Support 24x7 |

**Related reference:**

- “db2GetContacts API - Get the list of contacts to whom notification messages can be sent” in *Administrative API Reference*
- “CONTACTS administrative view – Retrieve list of contacts” in *Administrative SQL Routines and Views*

## GET DATABASE CONFIGURATION

Returns the values of individual entries in a specific database configuration file.

### Scope:

This command returns information only for the database partition on which it is executed.

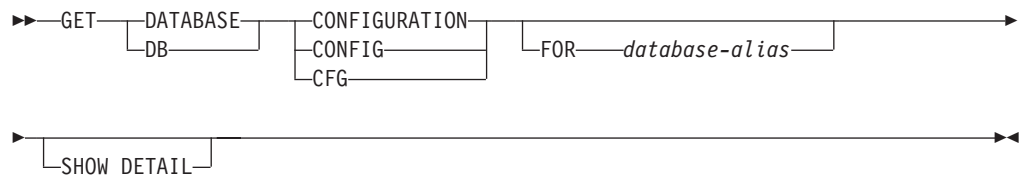
### Authorization:

None

### Required connection:

Instance. An explicit attachment is not required, but a connection to the database is required when using the SHOW DETAIL clause. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command.

### Command syntax:



### Command parameters:

#### FOR database-alias

Specifies the alias of the database whose configuration is to be displayed. You do not need to specify the alias if a connection to the database already exists.

#### SHOW DETAIL

Displays detailed information showing the current value of database configuration parameters as well as the value of the parameters the next time you activate the database. This option lets you see the result of dynamic changes to configuration parameters.

### Examples:

#### Notes:

1. Output on different platforms might show small variations reflecting platform-specific parameters.
2. Parameters with keywords enclosed by parentheses can be changed by the UPDATE DATABASE CONFIGURATION command.
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 mick
```

```
Database configuration release level = 0x0a00
```

## GET DATABASE CONFIGURATION

```

Database release level = 0x0a00

Database territory = en_US
Database code page = 819
Database code set = ISO8859-1
Database country/region code = 1
Database collating sequence = UNIQUE
Alternate collating sequence (ALT_COLLATE) =
Database page size = 4096
Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE

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
Default refresh age (DFT_REFRESH_AGE) = 0
Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
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 = YES

Log retain for recovery status = NO
User exit for logging status = NO

Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60
Data Links Write Token Init Expiry Intvl (DL_WT_IEXPINT) = 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) = MACO

Database heap (4KB) (DBHEAP) = 1200
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC
Catalog cache size (4KB) (CATALOGCACHE_SZ) = 64
Log buffer size (4KB) (LOGBUFSZ) = 8
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000
Buffer pool size (pages) (BUFFPAGE) = 1000
Max storage for lock list (4KB) (LOCKLIST) = 128

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 30000
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 128

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)
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

```

## GET DATABASE CONFIGURATION

```

Default prefetch size (pages) (DFT_PREFETCH_SZ) = AUTOMATIC

Track modified pages (TRACKMOD) = OFF

Default number of containers = 1
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32

Max number of active applications (MAXAPPLS) = AUTOMATIC
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/db2inst/db2inst
 /NODE0000/SQL00001
 /SQLOGDIR/

Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file =
Block log on disk full (BLK_LOG_DSK_FUL) = NO
Percent of max primary log space by transaction (MAX_LOG) = 0
Num. of active log files for 1 active UOW (NUM_LOG_SPAN) = 0

Group commit count (MINCOMMIT) = 1
Percent log file reclaimed before soft ckcpt (SOFTMAX) = 100
Log retain for recovery enabled (LOGRETAIN) = OFF
User exit for logging enabled (USEREXIT) = OFF

HADR database role = STANDARD
HADR local host name (HADR_LOCAL_HOST) =
HADR local service name (HADR_LOCAL_SVC) =
HADR remote host name (HADR_REMOTE_HOST) =
HADR remote service name (HADR_REMOTE_SVC) =
HADR instance name of remote server (HADR_REMOTE_INST) =
HADR timeout value (HADR_TIMEOUT) = 120
HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC

First log archive method (LOGARCHMETH1) = OFF
Options for logarchmeth1 (LOGARCHOPT1) =
Second log archive method (LOGARCHMETH2) = OFF
Options for logarchmeth2 (LOGARCHOPT2) =
Failover log archive path (FAILARCHPATH) =
Number of log archive retries on error (NUMARCHRETRY) = 5
Log archive retry Delay (secs) (ARCHRETRYDELAY) = 20
Vendor options (VENDOROPT) =

Auto restart enabled (AUTORESTART) = ON
Index re-creation time and redo index build (INDEXREC) = SYSTEM (RESTART)
Log pages during index build (LOGINDEXBUILD) = OFF
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

TSM management class (TSM_MGMTCLASS) =
TSM node name (TSM_NODENAME) =
TSM owner (TSM_OWNER) =
TSM password (TSM_PASSWORD) =

Automatic maintenance (AUTO_MAINT) = OFF
 Automatic database backup (AUTO_DB_BACKUP) = OFF
 Automatic table maintenance (AUTO_TBL_MAINT) = OFF
 Automatic runstats (AUTO_RUNSTATS) = OFF
 Automatic statistics profiling (AUTO_STATS_PROF) = OFF
 Automatic profile updates (AUTO_PROF_UPD) = OFF
 Automatic reorganization (AUTO_REORG) = OFF

```

## GET DATABASE CONFIGURATION

The following example shows a portion of the output of the command when you specify the SHOW DETAIL option. The value in the **Delayed Value** column is the value that will be applied the next time you start the instance.

| Database Configuration for Database mick |                   |                        |                      |  |
|------------------------------------------|-------------------|------------------------|----------------------|--|
| Description                              | Parameter         | Current Value          | Delayed Value        |  |
| Database configuration release level     |                   | = 0x0a00               |                      |  |
| Database release level                   |                   | = 0x0a00               |                      |  |
| Database territory                       |                   | = en_US                |                      |  |
| Database code page                       |                   | = 819                  |                      |  |
| Database code set                        |                   | = ISO8859-1            |                      |  |
| Database country/region code             |                   | = 1                    |                      |  |
| Database collating sequence              |                   | = UNIQUE               | UNIQUE               |  |
| Alternate collating sequence             | (ALT_COLLATE)     | =                      |                      |  |
| Database page size                       |                   | = 4096                 |                      |  |
| Dynamic SQL Query management             | (DYN_QUERY_MGMT)  | = DISABLE              | DISABLE              |  |
| Discovery support for this database      | (DISCOVER_DB)     | = ENABLE               | ENABLE               |  |
| Default query optimization class         | (DFT_QUERYOPT)    | = 5                    | 5                    |  |
| Degree of parallelism                    | (DFT_DEGREE)      | = 1                    | 1                    |  |
| Continue upon arithmetic exceptions      | (DFT_SQLMATHWARN) | = NO                   | NO                   |  |
| Default refresh age                      | (DFT_REFRESH_AGE) | = 0                    | 0                    |  |
| Default maintained table types for opt   | (DFT_MTTB_TYPES)  | = SYSTEM               | SYSTEM               |  |
| Number of frequent values retained       | (NUM_FREQVALUES)  | = 10                   | 10                   |  |
| Number of quantiles retained             | (NUM_QUANTILES)   | = 20                   | 20                   |  |
| Backup pending                           |                   | = NO                   |                      |  |
| Database is consistent                   |                   | = YES                  |                      |  |
| Rollforward pending                      |                   | = NO                   |                      |  |
| Restore pending                          |                   | = NO                   |                      |  |
| Multi-page file allocation enabled       |                   | = YES                  |                      |  |
| Log retain for recovery status           |                   | = NO                   |                      |  |
| User exit for logging status             |                   | = NO                   |                      |  |
| Data Links Token Expiry Interval (sec)   | (DL_EXPINT)       | = 60                   | 60                   |  |
| Data Links Write Token Init Expiry Intvl | (DL_WT_IEXPINT)   | = 60                   | 60                   |  |
| Data Links Number of Copies              | (DL_NUM_COPIES)   | = 1                    | 1                    |  |
| Data Links Time after Drop (days)        | (DL_TIME_DROP)    | = 1                    | 1                    |  |
| Data Links Token in Uppercase            | (DL_UPPER)        | = NO                   | NO                   |  |
| Data Links Token Algorithm               | (DL_TOKEN)        | = MACO                 | MACO                 |  |
| Database heap (4KB)                      | (DBHEAP)          | = 1200                 | 1200                 |  |
| Size of database shared memory (4KB)     | (DATABASE_MEMORY) | = AUTOMATIC<br>(11516) | AUTOMATIC<br>(11516) |  |
| Catalog cache size (4KB)                 | (CATALOGCACHE_SZ) | = 64                   | 64                   |  |
| Log buffer size (4KB)                    | (LOGBUFSZ)        | = 8                    | 8                    |  |
| Utilities heap size (4KB)                | (UTIL_HEAP_SZ)    | = 5000                 | 5000                 |  |
| Buffer pool size (pages)                 | (BUFFPAGE)        | = 1000                 | 1000                 |  |
| Max storage for lock list (4KB)          | (LOCKLIST)        | = 128                  | 128                  |  |
| Max size of appl. group mem set (4KB)    | (APPGROUP_MEM_SZ) | = 30000                | 30000                |  |
| Percent of mem for appl. group heap      | (GROUPHEAP_RATIO) | = 70                   | 70                   |  |
| Max appl. control heap size (4KB)        | (APP_CTL_HEAP_SZ) | = 128                  | 128                  |  |
| Sort heap thres for shared sorts (4KB)   | (SHEAPTHRES_SHR)  | = (SHEAPTHRES)         | (SHEAPTHRES)         |  |
| Sort list heap (4KB)                     | (SORTHEAP)        | = 256                  | 256                  |  |
| SQL statement heap (4KB)                 | (STMTHEAP)        | = 2048                 | 2048                 |  |
| Default application heap (4KB)           | (APPLHEAPSZ)      | = 128                  | 128                  |  |
| Package cache size (4KB)                 | (PCKCACHESZ)      | = (MAXAPPLS*8)         | (MAXAPPLS*8)         |  |
| Statistics heap size (4KB)               | (STAT_HEAP_SZ)    | = 4384                 | 4384                 |  |
| Interval for checking deadlock (ms)      | (DLCHKTIME)       | = 10000                | 10000                |  |
| Percent. of lock lists per application   | (MAXLOCKS)        | = 10                   | 10                   |  |

## GET DATABASE CONFIGURATION

```

Lock timeout (sec) (LOCKTIMEOUT) = -1 -1

Changed pages threshold (CHNGPGS_THRESH) = 60 60
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 1 1
Number of I/O servers (NUM_IOSERVERS) = 3 3
Index sort flag (INDEXSORT) = YES YES
Sequential detect flag (SEQDETECT) = YES YES
Default prefetch size (pages) (DFT_PREFETCH_SZ) = AUTOMATIC AUTOMATIC

Track modified pages (TRACKMOD) = NO NO

Default number of containers = 1 1
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32 32

Max number of active applications (MAXAPPLS) = AUTOMATIC AUTOMATIC
 (40) (40)
Average number of active applications (AVG_APPLS) = 1 1
Max DB files open per application (MAXFILOP) = 64 64

Log file size (4KB) (LOGFILSIZ) = 1000 1000
Number of primary log files (LOGPRIMARY) = 3 3
Number of secondary log files (LOGSECOND) = 2 2
Changed path to log files (NEWLOGPATH) =
Path to log files = home/db2inst /home
 /db2inst /db2inst
 /NODE0000 /db2inst
 /SQL00001 /NODE0000
 /SQLOGDIR/ /SQL00001
 /SQLOGDIR/

Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file =
Block log on disk full (BLK_LOG_DSK_FUL) = NO NO
Percent of max primary log space by transaction (MAX_LOG) = 0 0
Num. of active log files for 1 active UOW (NUM_LOG_SPAN) = 0 0
Group commit count (MINCOMMIT) = 1 1
Percent log file reclaimed before soft chkpt (SOFTMAX) = 100 100
Log retain for recovery enabled (LOGRETAIN) = OFF OFF
User exit for logging enabled (USEREXIT) = OFF OFF

HADR database role = STANDARD STANDARD
HADR local host name (HADR_LOCAL_HOST) =
HADR local service name (HADR_LOCAL_SVC) =
HADR remote host name (HADR_REMOTE_HOST) =
HADR remote service name (HADR_REMOTE_SVC) =
HADR instance name of remote server (HADR_REMOTE_INST) =
HADR timeout value (HADR_TIMEOUT) = 120 120
HADR log write synchronization mode (HADR_SYNCMODE) = NEARSYNC NEARSYNC

First log archive method (LOGARCHMETH1) = OFF OFF
Options for logarchmeth1 (LOGARCHOPT1) =
Second log archive method (LOGARCHMETH2) = OFF OFF
Options for logarchmeth2 (LOGARCHOPT2) =
Failover log archive path (FAILARCHPATH) =
Number of log archive retries on error (NUMARCHRETRY) = 5 5
Log archive retry Delay (secs) (ARCHRETRYDELAY) = 20 20
Vendor options (VENDOROPT) =
Auto restart enabled (AUTORESTART) = ON ON
Index re-creation time and redo index build (INDEXREC) = SYSTEM SYSTEM
 (RESTART) (RESTART)

Log pages during index build (LOGINDEXBUILD) = OFF OFF
Default number of loadrec sessions (DFT_LOADREC_SES) = 1 1
Number of database backups to retain (NUM_DB_BACKUPS) = 12 12
Recovery history retention (days) (REC_HIS_RETENTN) = 366 366

TSM management class (TSM_MGMTCLASS) =
TSM node name (TSM_NODENAME) =

```

## GET DATABASE CONFIGURATION

|                                |                     |     |
|--------------------------------|---------------------|-----|
| TSM owner                      | (TSM_OWNER) =       |     |
| TSM password                   | (TSM_PASSWORD) =    |     |
| Automatic maintenance          | (AUTO_MAINT) =      | OFF |
| Automatic database backup      | (AUTO_DB_BACKUP) =  | OFF |
| Automatic table maintenance    | (AUTO_TBL_MAINT) =  | OFF |
| Automatic runstats             | (AUTO_RUNSTATS) =   | OFF |
| Automatic statistics profiling | (AUTO_STATS_PROF) = | OFF |
| Automatic profile updates      | (AUTO_PROF_UPD) =   | OFF |
| Automatic reorganization       | (AUTO_REORG) =      | OFF |

### 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 the RESET DATABASE CONFIGURATION command.

To retrieve information from all database partitions, use the SYSIBMADM.DBCFG administrative view.

### Related tasks:

- “Changing node and database configuration files” in *Administration Guide: Implementation*
- “Configuring DB2 with configuration parameters” in *Performance Guide*

### Related reference:

- “RESET DATABASE CONFIGURATION ” on page 667
- “UPDATE DATABASE CONFIGURATION ” on page 772
- “db2CfgGet API - Get the database manager or database configuration parameters” in *Administrative API Reference*
- “Configuration parameters summary” in *Performance Guide*
- “DBCFG administrative view – Retrieve database configuration parameter information” in *Administrative SQL Routines and Views*



## GET DATABASE MANAGER CONFIGURATION

Returns the values of individual entries in the database manager configuration file.

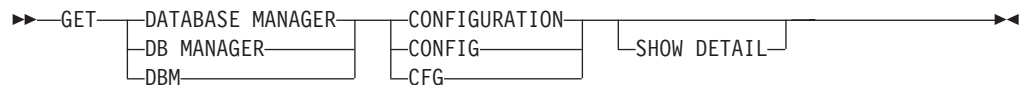
### Authorization:

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. The SHOW DETAIL clause requires an instance attachment.

### Command syntax:



### Command parameters:

#### SHOW DETAIL

Displays detailed information showing the current value of database manager configuration parameters as well as the value of the parameters the next time you start the database manager. This option lets you see the result of dynamic changes to configuration parameters.

### Examples:

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 = 0x0a00
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 installation path (JDK_PATH) = /usr/java131
Diagnostic error capture level (DIAGLEVEL) = 3
Notify Level (NOTIFYLEVEL) = 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

```

## GET DATABASE MANAGER CONFIGURATION

```

Statement (DFT_MON_STMT) = OFF
Table (DFT_MON_TABLE) = OFF
Timestamp (DFT_MON_TIMESTAMP) = ON
Unit of work (DFT_MON_UOW) = OFF
Monitor health of instance and databases (HEALTH_MON) = ON

SYSADM group name (SYSADM_GROUP) =
SYSCTRL group name (SYSCTRL_GROUP) =
SYSMAINT group name (SYSMAINT_GROUP) =
SYSMON group name (SYSMON_GROUP) =

Client Userid-Password Plugin (CLNT_PW_PLUGIN) =
Client Kerberos Plugin (CLNT_KRB_PLUGIN) =
Group Plugin (GROUP_PLUGIN) =
GSS Plugin for Local Authorization (LOCAL_GSSPLUGIN) =
Server Plugin Mode (SRV_PLUGIN_MODE) = UNFENCED
Server List of GSS Plugins (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin (SRVCON_PW_PLUGIN) =
Server Connection Authentication (SRVCON_AUTH) = NOT_SPECIFIED
Database manager authentication (AUTHENTICATION) = SERVER
Cataloging allowed without authority (CATALOG_NOAUTH) = YES
Trust all clients (TRUST_ALLCLNTS) = YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication (FED_NOAUTH) = NO

Default database path (DFTDBPATH) = /home/db2inst

Database monitor heap size (4KB) (MON_HEAP_SZ) = 90
Java Virtual Machine heap size (4KB) (JAVA_HEAP_SZ) = 512
Audit buffer size (4KB) (AUDIT_BUF_SZ) = 0
Size of instance shared memory (4KB) (INSTANCE_MEMORY) = AUTOMATIC
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

Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

Priority of agents (AGENTPRI) = SYSTEM
Max number of existing agents (MAXAGENTS) = 200
Agent pool size (NUM_POOLAGENTS) = 100(calculated)
Initial number of agents in pool (NUM_INITAGENTS) = 0
Max number of coordinating agents (MAX_COORDAGENTS) = MAXAGENTS
Max no. of concurrent coordinating agents (MAXCAGENTS) = MAX_COORDAGENTS
Max number of client connections (MAX_CONNECTIONS) = MAX_COORDAGENTS

Keep fenced process (KEEPFENCED) = YES
Number of pooled fenced processes (FENCED_POOL) = MAX_COORDAGENTS
Initial number of fenced processes (NUM_INITFENCED) = 0

Index re-creation time and redo index build (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) =
Discovery mode (DISCOVER) = SEARCH
Discover server instance (DISCOVER_INST) = ENABLE

Maximum query degree of parallelism (MAX_QUERYDEGREE) = ANY
Enable intra-partition parallelism (INTRA_PARALLEL) = NO

```

## GET DATABASE MANAGER CONFIGURATION

No. of int. communication buffers(4KB) (FCM\_NUM\_BUFFERS) = AUTOMATIC  
 No. of int. communication channels (FCM\_NUM\_CHANNELS) = AUTOMATIC

The following output sample shows the information displayed when you specify the WITH DETAIL option. The value that appears in the **Delayed Value** is the value that will be in effect the next time you start the database manager instance.

```

 Database Manager Configuration
Node type = Database Server with local clients
Description Parameter Current Value Delayed
 Parameter Value Value
Database manager configuration release level = 0x0a00

CPU speed (millisec/instruction) (CPUSPEED) = 4.000000e 4.000000e
 -05 -05
Max number of concurrently active databases (NUMDB) = 8 8
Data Links support (DATALINKS) = NO NO
Federated Database System Support (FEDERATED) = NO NO
Transaction processor monitor name (TP_MON_NAME) =

Default charge-back account (DFT_ACCOUNT_STR) =

Java Development Kit installation path (JDK_PATH) = /wsdb/v81 /usr
 /bldsupp /java131
 /AIX/jdk1.3.1

Diagnostic error capture level (DIAGLEVEL) = 3 3
Notify Level (NOTIFYLEVEL) = 3 3
Diagnostic data directory path (DIAGPATH) =

Default database monitor switches
 Buffer pool (DFT_MON_BUFPOOL) = OFF OFF
 Lock (DFT_MON_LOCK) = OFF OFF
 Sort (DFT_MON_SORT) = OFF OFF
 Statement (DFT_MON_STMT) = OFF OFF
 Table (DFT_MON_TABLE) = OFF OFF
 Timestamp (DFT_MON_TIMESTAMP) = ON ON
 Unit of work (DFT_MON_UOW) = OFF OFF
Monitor health of instance and databases (HEALTH_MON) = ON ON

SYSADM group name (SYSADM_GROUP) = BUILD
SYSCTRL group name (SYSCTRL_GROUP) =
SYSMAINT group name (SYSMAINT_GROUP) =
SYSMON group name (SYSMON_GROUP) =

Client Userid-Password Plugin (CLNT_PW_PLUGIN) =
Client Kerberos Plugin (CLNT_KRB_PLUGIN) =
Group Plugin (GROUP_PLUGIN) =
GSS Plugin for Local Authorization (LOCAL_GSSPLUGIN) =
Server Plugin Mode (SRV_PLUGIN_MODE) = UNFENCED UNFENCED
Server List of GSS Plugins (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin (SRVCON_PW_PLUGIN) =
Server Connection Authentication (SRVCON_AUTH) = NOT_ NOT_
 SPECIFIED SPECIFIED
Database manager authentication (AUTHENTICATION) = SERVER SERVER
Cataloging allowed without authority (CATALOG_NOAUTH) = YES YES
Trust all clients (TRUST_ALLCLNTS) = YES YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT CLIENT
Bypass federated authentication (FED_NOAUTH) = NO NO

Default database path (DFTDBPATH) = /home /home
 /db2inst /db2inst

```

## GET DATABASE MANAGER CONFIGURATION

|                                             |                                         |                     |
|---------------------------------------------|-----------------------------------------|---------------------|
| Database monitor heap size (4KB)            | (MON_HEAP_SZ) = 90                      | 90                  |
| Java Virtual Machine heap size (4KB)        | (JAVA_HEAP_SZ) = 512                    | 512                 |
| Audit buffer size (4KB)                     | (AUDIT_BUF_SZ) = 0                      | 0                   |
| Size of instance shared memory (4KB)        | (INSTANCE_MEMORY) = AUTOMATIC<br>(5386) | AUTOMATIC<br>(20)   |
| Backup buffer default size (4KB)            | (BACKBUFSZ) = 1024                      | 1024                |
| Restore buffer default size (4KB)           | (RESTBUFSZ) = 1024                      | 1024                |
| Sort heap threshold (4KB)                   | (SHEAPTHRES) = 20000                    | 20000               |
| Directory cache support                     | (DIR_CACHE) = YES                       | YES                 |
| Application support layer heap size (4KB)   | (ASLHEAPSZ) = 15                        | 15                  |
| Max requester I/O block size (bytes)        | (RQRIOBLK) = 32767                      | 32767               |
| Query heap size (4KB)                       | (QUERY_HEAP_SZ) = 1000                  | 1000                |
| Workload impact by throttled utilities      | (UTIL_IMPACT_LIM) = 10                  | 10                  |
| Priority of agents                          | (AGENTPRI) = SYSTEM                     | SYSTEM              |
| Max number of existing agents               | (MAXAGENTS) = 200                       | 200                 |
| Agent pool size                             | (NUM_POOLAGENTS) = 100                  | 100<br>(calculated) |
| Initial number of agents in pool            | (NUM_INITAGENTS) = 0                    | 0                   |
| Max number of coordinating agents           | (MAX_COORDAGENTS) = 200                 | MAXAGENTS           |
| Max no. of concurrent coordinating agents   | (MAXCAGENTS) = 200                      | MAX_COORDAGENTS     |
| Max number of client connections            | (MAX_CONNECTIONS) = 200                 | MAX_COORDAGENTS     |
| Keep fenced process                         | (KEEPFENCED) = YES                      | YES                 |
| Number of pooled fenced processes           | (FENCED_POOL) = MAX<br>COORDAGENTS      | MAX<br>COORDAGENTS  |
| Initial number of fenced processes          | (NUM_INITFENCED) = 0                    | 0                   |
| Index re-creation time and redo index build | (INDEXREC) = RESTART                    | RESTART             |
| Transaction manager database name           | (TM_DATABASE) = 1ST_CONN                | 1ST_CONN            |
| Transaction resync interval (sec)           | (RESYNC_INTERVAL) = 180                 | 180                 |
| SPM name                                    | (SPM_NAME) =                            |                     |
| SPM log size                                | (SPM_LOG_FILE_SZ) = 256                 | 256                 |
| SPM resync agent limit                      | (SPM_MAX_RESYNC) = 20                   | 20                  |
| SPM log path                                | (SPM_LOG_PATH) =                        |                     |
| TCP/IP Service name                         | (SVCENAME) =                            |                     |
| Discovery mode                              | (DISCOVER) = SEARCH                     | SEARCH              |
| Discover server instance                    | (DISCOVER_INST) = ENABLE                | ENABLE              |
| Maximum query degree of parallelism         | (MAX_QUERYDEGREE) = ANY                 | ANY                 |
| Enable intra-partition parallelism          | (INTRA_PARALLEL) = NO                   | NO                  |
| No. of int. communication buffers(4KB)      | (FCM_NUM_BUFFERS) = AUTOMATIC           | AUTOMATIC           |
| No. of int. communication channels          | (FCM_NUM_CHANNELS) = AUTOMATIC          | AUTOMATIC           |

### 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 the **RESET DATABASE MANAGER CONFIGURATION** command.
- The AUTOMATIC values indicated on **get database manager configuration show detail** for FCM\_NUM\_BUFFERS and FCM\_NUM\_CHANNELS are the initial values at

## GET DATABASE MANAGER CONFIGURATION

instance startup time and do not reflect any automatic increasing/decreasing that might have occurred during runtime.

### Related tasks:

- “Changing node and database configuration files” in *Administration Guide: Implementation*
- “Configuring DB2 with configuration parameters” in *Performance Guide*

### Related reference:

- “RESET DATABASE MANAGER CONFIGURATION ” on page 669
- “UPDATE DATABASE MANAGER CONFIGURATION ” on page 775
- “db2CfgGet API - Get the database manager or database configuration parameters” in *Administrative API Reference*
- “Configuration parameters summary” in *Performance Guide*
- “DBMCFG administrative view – Retrieve database manager configuration parameter information” in *Administrative SQL Routines and Views*

## 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. 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*
- *sysmon*

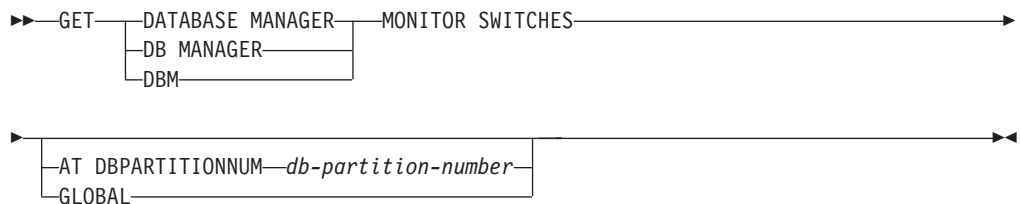
**Required connection:**

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:**



**Command parameters:**

**AT DBPARTITIONNUM db-partition-number**

Specifies the database partition for which the status of the database manager monitor switches is to be displayed.

**GLOBAL**

Returns an aggregate result for all database partitions in a partitioned database system.

**Examples:**

The following is sample output from GET DATABASE MANAGER MONITOR SWITCHES:

## GET DATABASE MANAGER MONITOR SWITCHES

DBM System Monitor Information Collected

```
Switch list for db partition number 1
Buffer Pool Activity Information (BUFFERPOOL) = ON 06-11-2003 10:11:01.738377
Lock Information (LOCK) = OFF
Sorting Information (SORT) = ON 06-11-2003 10:11:01.738400
SQL Statement Information (STATEMENT) = OFF
Table Activity Information (TABLE) = OFF
Take Timestamp Information (TIMESTAMP) = ON 06-11-2003 10:11:01.738525
Unit of Work Information (UOW) = ON 06-11-2003 10:11:01.738353
```

### Usage notes:

The recording switches BUFFERPOOL, LOCK, SORT, STATEMENT, TABLE, and UOW are off by default, but can be switched on using the UPDATE MONITOR SWITCHES command. If any of these switches are on, this command also displays the time stamp for when the switch was turned on.

The recording switch TIMESTAMP is on by default, but can be switched off using UPDATE MONITOR SWITCHES. When this switch is on the system issues timestamp calls when collecting information for timestamp monitor elements. Examples of these elements are:

- agent\_sys\_cpu\_time
- agent\_usr\_cpu\_time
- appl\_con\_time
- con\_elapsed\_time
- con\_response\_time
- conn\_complete\_time
- db\_conn\_time
- elapsed\_exec\_time
- gw\_comm\_error\_time
- gw\_con\_time
- gw\_exec\_time
- host\_response\_time
- last\_backup
- last\_reset
- lock\_wait\_start\_time
- network\_time\_bottom
- network\_time\_top
- prev\_uow\_stop\_time
- rf\_timestamp
- ss\_sys\_cpu\_time
- ss\_usr\_cpu\_time
- status\_change\_time
- stmt\_elapsed\_time
- stmt\_start
- stmt\_stop
- stmt\_sys\_cpu\_time
- stmt\_usr\_cpu\_time
- uow\_elapsed\_time

## GET DATABASE MANAGER MONITOR SWITCHES

- uow\_start\_time
- uow\_stop\_time

If the **TIMESTAMP** switch is off, timestamp operating system calls are not issued to determine these elements and these elements will contain zero. Turning this switch off becomes important as CPU utilization approaches 100%; when this occurs, the CPU time required for issuing timestamps increases dramatically.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword **NODE** can be substituted for **DBPARTITIONNUM**.

### Related concepts:

- “System monitor switches” in *System Monitor Guide and Reference*

### Related tasks:

- “Setting monitor switches from a client application” in *System Monitor Guide and Reference*

### Related reference:

- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*
- “GET SNAPSHOT ” on page 487
- “GET MONITOR SWITCHES ” on page 478
- “RESET MONITOR ” on page 671
- “UPDATE MONITOR SWITCHES ” on page 782
- “SNAPSWITCHES administrative view and SNAP\_GET\_SWITCHES table function – Retrieve database snapshot switch state information” in *Administrative SQL Routines and Views*



---

## GET DESCRIPTION FOR HEALTH INDICATOR

Returns a description for the specified health indicator. A Health Indicator measures the healthiness of a particular state, capacity, or behavior of the database system. The state defines whether or not the database object or resource is operating normally.

### Authorization:

None.

### Required connection:

Instance. If there is no instance attachment, a default instance attachment is created.

To obtain a snapshot of a remote instance, it is necessary to first attach to that instance.

### Command syntax:

►►—GET DESCRIPTION FOR HEALTH INDICATOR—*shortname*—————◄◄

### Command parameters:

**HEALTH INDICATOR** *shortname*

The name of the health indicator for which you would like to retrieve the description. Health indicator names consist of a two- or three-letter object identifier followed by a name which describes what the indicator measures. For example:

```
db.sort_privmem_util
```

### Examples:

The following is sample output from the GET DESCRIPTION FOR HEALTH INDICATOR command.

```
GET DESCRIPTION FOR HEALTH INDICATOR db2.sort_privmem_util
```

```
DESCRIPTION FOR db2.sort_privmem_util
```

Sorting is considered healthy if there is sufficient heap space in which to perform sorting and sorts do not overflow unnecessarily. This indicator tracks the utilization of the private sort memory. If `db2.sort_heap_allocated` (system monitor data element)  $\geq$  `SHEAPTHRES` (DBM configuration parameter), sorts may not be getting full sort heap as defined by the `SORTHEAP` parameter and an alert may be generated. The indicator is calculated using the formula:  $(db2.sort\_heap\_allocated / SHEAPTHRES) * 100$ . The Post Threshold Sorts snapshot monitor element measures the number of sorts that have requested heaps after the sort heap threshold has been exceeded. The value of this indicator, shown in the Additional Details, indicates the degree of severity of the problem for this health indicator. The Maximum Private Sort Memory Used snapshot monitor element maintains a private sort memory high-water mark for the instance. The value of this indicator, shown in the Additional Information, indicates the maximum amount of private sort memory that has been in use at any one point in time since the instance was last recycled. This value can be used to help determine an appropriate value for `SHEAPTHRES`.

### Related reference:

## GET DESCRIPTION FOR HEALTH INDICATOR

- “Health indicators” in *System Monitor Guide and Reference*
- “HEALTH\_GET\_IND\_DEFINITION table function – Retrieve health indicator definitions” in *Administrative SQL Routines and Views*

## GET HEALTH NOTIFICATION CONTACT LIST

Returns the list of contacts and contact groups that are notified about the health of an instance. A contact list consists of e-mail addresses or pager Internet addresses of individuals who are to be notified when non-normal health conditions are present for an instance or any of its database objects.

**Authorization:**

None.

**Required Connection:**

Instance. An explicit attachment is not required.

**Command Syntax:**



**Command Parameters:**

None.

**Examples:**

Issuing the command GET NOTIFICATION LIST results in a report similar to the following:

| Name      | Type          |
|-----------|---------------|
| -----     |               |
| Joe Brown | Contact       |
| Support   | Contact group |

**Related reference:**

- “db2GetHealthNotificationList API - Get the list of contacts to whom health alert notifications can be sent” in *Administrative API Reference*
- “NOTIFICATIONLIST administrative view – Retrieve contact list for health notification” in *Administrative SQL Routines and Views*

## GET HEALTH SNAPSHOT

Retrieves the health status information for the database manager and its databases. The information returned represents a snapshot of the health state at the time the command was issued.

**Scope:**

In a partitioned database environment, this command can be invoked from any database partition defined in the `db2nodes.cfg` file. By default it acts on the database partition from which it was invoked. If you use the GLOBAL option, it will extract consolidated information from all of the database partitions.

**Authorization:**

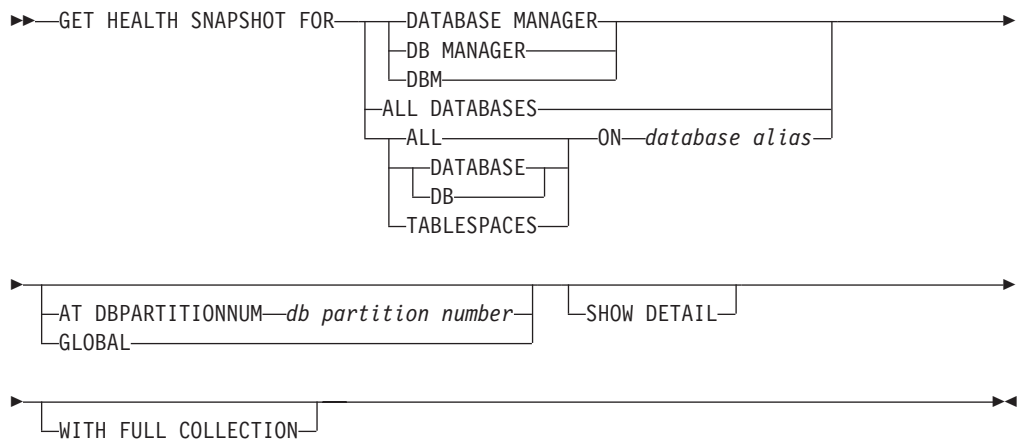
None.

**Required connection:**

Instance. If there is no instance attachment, a default instance attachment is created.

To obtain a snapshot of a remote instance, it is necessary to first attach to that instance.

**Command syntax:**



**Command parameters:**

**DATABASE MANAGER**

Provides statistics for the active database manager instance.

**ALL DATABASES**

Provides health states for all active databases on the current database partition.

**ALL ON database-alias**

Provides health states and information about all table spaces and buffer pools for a specified database.

**DATABASE ON database-alias**

**TABLESPACES ON database-alias**

Provides information about table spaces for a specified database.

**AT DBPARTITIONNUM db-partition-number**

Returns results for the database partition specified.

**GLOBAL**

Returns an aggregate result for all database partitions in a partitioned database system.

**SHOW DETAIL**

Specifies that the output should include the historical data for each health monitor data element in the form of {(Timestamp, Value, Formula)}, where the bracketed parameters (Timestamp, Value, Formula), will be repeated for each history record that is returned. For example,

```
(03-19-2002 13:40:24.138865,50,((1-(4/8))*100)),
(03-19-2002 13:40:13.1386300,50,((1-(4/8))*100)),
(03-19-2002 13:40:03.1988858,0,((1-(3/3))*100))
```

Collection object history is returned for all collection objects in ATTENTION or AUTOMATE FAILED state.

The SHOW DETAIL option also provides additional contextual information that can be useful to understanding the value and alert state of the associated Health Indicator. For example, if the table space storage utilization Health Indicator is being used to determine how full the table space is, the rate at which the table space is growing will also be provided by SHOW DETAIL.

**WITH FULL COLLECTION**

Specifies that full collection information for all collection state-based health indicators is to be returned. This option considers both the name and size filter criteria. If a user requests a health snapshot with full collection, the report will show all tables that meet the name and size criteria in the policy. This can be used to validate which tables will be evaluated in a given refresh cycle. The output returned when this option is specified is for collection objects in NORMAL, AUTOMATED, ATTENTION, or AUTOMATE FAILED state. This option can be specified in conjunction with the SHOW DETAIL option.

Without this option, only tables that have been evaluated for automatic reorganization and require manual intervention (i.e. manual reorg or automation failed) will be displayed in a get health snapshot report.

**Examples:**

The following is typical output resulting from a request for database manager information:

```
D:\>DB2 GET HEALTH SNAPSHOT FOR DBM
```

Database Manager Health Snapshot

```
Node name =
Node type = Enterprise Server Edition
 with local and remote clients
Instance name = DB2
Snapshot timestamp = 02/17/2004 12:39:44.818949

Number of database partitions in DB2 instance = 1
Start Database Manager timestamp = 02/17/2004 12:17:21.000119
```

## GET HEALTH SNAPSHOT

Instance highest severity alert state = Normal

### Health Indicators:

Indicator Name = db2.db2\_op\_status  
Value = 0  
Evaluation timestamp = 02/17/2004 12:37:23.393000  
Alert state = Normal

Indicator Name = db2.sort\_privmem\_util  
Value = 0  
Unit = %  
Evaluation timestamp = 02/17/2004 12:37:23.393000  
Alert state = Normal

Indicator Name = db2.mon\_heap\_util  
Value = 6  
Unit = %  
Evaluation timestamp = 02/17/2004 12:37:23.393000  
Alert state = Normal

### Related tasks:

- “Capturing a database health snapshot using the CLP” in *System Monitor Guide and Reference*

### Related reference:

- “Health monitor CLP commands” in *System Monitor Guide and Reference*
- “Health monitor interface mappings to logical data groups” in *System Monitor Guide and Reference*
- “Health monitor sample output” in *System Monitor Guide and Reference*

---

## 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

**Related tasks:**

- “Setting the current instance environment variables” in *Administration Guide: Implementation*

**Related reference:**

- “sqlqins API - Get current instance” in *Administrative API Reference*

## 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 the GET DBM MONITOR SWITCHES command.

**Authorization:**

One of the following:

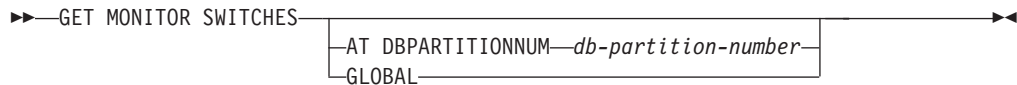
- *sysadm*
- *sysctrl*
- *sysmaint*
- *sysmon*

**Required connection:**

Instance. If there is no instance attachment, a default instance attachment is created.

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:**



**Command parameters:**

**AT DBPARTITIONNUM db-partition-number**

Specifies the database partition for which the status of the monitor switches is to be displayed.

**GLOBAL**

Returns an aggregate result for all database partitions in a partitioned database system.

**Examples:**

The following is sample output from GET MONITOR SWITCHES:

```

 Monitor Recording Switches

 Switch list for db partition number 1
 Buffer Pool Activity Information (BUFFERPOOL) = ON 02-20-2003 16:04:30.070073
 Lock Information (LOCK) = OFF
 Sorting Information (SORT) = OFF
 SQL Statement Information (STATEMENT) = ON 02-20-2003 16:04:30.070073
 Table Activity Information (TABLE) = OFF
 Take Timestamp Information (TIMESTAMP) = ON 02-20-2003 16:04:30.070073
 Unit of Work Information (UOW) = ON 02-20-2003 16:04:30.070073

```

**Usage notes:**



The recording switch `TIMESTAMP` is on by default, but can be switched off using `UPDATE MONITOR SWITCHES`. When this switch is on the system issues timestamp calls when collecting information for timestamp monitor elements.

The recording switch `TIMESTAMP` is on by default, but can be switched off using `UPDATE MONITOR SWITCHES`. If this switch is off, this command also displays the time stamp for when the switch was turned off. When this switch is on the system issues timestamp calls when collecting information for timestamp monitor elements. Examples of these elements are:

- `agent_sys_cpu_time`
- `agent_usr_cpu_time`
- `appl_con_time`
- `con_elapsed_time`
- `con_response_time`
- `conn_complete_time`
- `db_conn_time`
- `elapsed_exec_time`
- `gw_comm_error_time`
- `gw_con_time`
- `gw_exec_time`
- `host_response_time`
- `last_backup`
- `last_reset`
- `lock_wait_start_time`
- `network_time_bottom`
- `network_time_top`
- `prev_uow_stop_time`
- `rf_timestamp`
- `ss_sys_cpu_time`
- `ss_usr_cpu_time`
- `status_change_time`
- `stmt_elapsed_time`
- `stmt_start`
- `stmt_stop`
- `stmt_sys_cpu_time`
- `stmt_usr_cpu_time`
- `uow_elapsed_time`
- `uow_start_time`
- `uow_stop_time`

If the `TIMESTAMP` switch is off, timestamp operating system calls are not issued to determine these elements and these elements will contain zero. Turning this switch off becomes important as CPU utilization approaches 100%; when this occurs, the CPU time required for issuing timestamps increases dramatically.

### **Compatibilities:**

For compatibility with versions earlier than Version 8:

## GET MONITOR SWITCHES

- The keyword NODE can be substituted for DBPARTITIONNUM.

### **Related concepts:**

- “System monitor switches” in *System Monitor Guide and Reference*

### **Related tasks:**

- “Setting monitor switches from a client application” in *System Monitor Guide and Reference*

### **Related reference:**

- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*
- “GET SNAPSHOT ” on page 487
- “GET DATABASE MANAGER MONITOR SWITCHES ” on page 468
- “RESET MONITOR ” on page 671
- “UPDATE MONITOR SWITCHES ” on page 782

## GET RECOMMENDATIONS FOR HEALTH INDICATOR

Returns descriptions of recommendations for improving the health of the aspect of the database system that is monitored by the specified health indicator. Recommendations can be returned for a health indicator that is in an alert state on a specific object, or the full set of recommendations for a given health indicator can be queried.

### Scope:

In a partitioned database environment, this command can be invoked from any database partition defined in the `db2nodes.cfg` file. It acts only on that database partition unless the `GLOBAL` parameter is specified.

### Authorization:

None.

### Required connection:

Instance. If there is no instance attachment, a default instance attachment is created. To retrieve recommendations for a remote instance, it is necessary to first attach to that instance.

### Command syntax:

```

>> GET RECOMMENDATIONS FOR HEALTH INDICATOR health-indicator-name
FOR DBM
 TABLESPACE tblspacename ON database-alias
 CONTAINER containername FOR TABLESPACE tblspacename
 DATABASE
AT DBPARTITIONNUM db-partition-number
GLOBAL

```

### Command parameters:

#### HEALTH INDICATOR *health-indicator-name*

The name of the health indicator for which you would like to retrieve the recommendations. Health indicator names consist of a two- or three-letter object identifier followed by a name that describes what the indicator measures.

**DBM** Returns recommendations for a database manager health indicator that has entered an alert state.

#### TABLESPACE

Returns recommendation for a health indicator that has entered an alert state on the specified table space and database.

#### CONTAINER

Returns recommendation for a health indicator that has entered an alert state on the specified container in the specified table space and database.

#### DATABASE

Returns recommendations for a health indicator that has entered an alert state on the specified database.

## GET RECOMMENDATIONS FOR HEALTH INDICATOR

**ON *database-alias***  
Specifies a database.

**AT DBPARTITIONNUM**  
Specifies the database partition number at which the health indicator has entered an alert state. If a database partition number is not specified and GLOBAL is not specified, the command will return information for the currently connected database partition.

**GLOBAL**  
Retrieves recommendations for the specified health indicator across all database partitions. In cases where the recommendations are the same on different database partitions, those recommendations are returned as a single set of recommendations that solve the health indicator on the affected database partitions.

### Examples:

```
db2 get recommendations for health indicator db.db_heap_util
for database on sample
```

Problem:

|                        |                       |
|------------------------|-----------------------|
| Indicator Name         | = db.db_heap_util     |
| Value                  | = 42                  |
| Evaluation timestamp   | = 11/25/2003 19:04:54 |
| Alert state            | = Alarm               |
| Additional information | =                     |

Recommendations:

Recommendation: Increase the database heap size.  
Rank: 1

Increase the database configuration parameter dbheap sufficiently to move utilization to normal operating levels. To increase the value, set the new value of dbheap to be equal to  $(pool\_cur\_size / (4096 * U))$  where U is the desired utilization rate. For example, if your desired utilization rate is 60% of the warning threshold level, which you have set at 75%, then  $U = 0.6 * 0.75 = 0.45$  (or 45%).

Take one of the following actions:

Execute the following scripts at the DB2 server:

```
CONNECT TO SAMPLE;
UPDATE DB CFG USING DBHEAP 149333;
CONNECT_RESET;
```

Launch DB2 tool: Database Configuration Window

The Database Configuration window can be used to view and update database configuration parameters.

To open the Database Configuration window:

1. From the Control Center, expand the object tree until you find the databases folder.
2. Click the databases folder. Any existing database are displayed in the contents pane on the right side of the window.
3. Right-click the database that you want in the contents pane, and click Configure Parameters in the pop-up menu. The Database Configuration window opens.

On the Performance tab, update the database heap size parameter as

## GET RECOMMENDATIONS FOR HEALTH INDICATOR

suggested and click OK to apply the update.

Recommendation: Investigate memory usage of database heap.  
Rank: 2

There is one database heap per database and the database manager uses it on behalf of all applications connected to the database. The data area is expanded as needed up to the maximum specified by dbheap.

For more information on the database heap, refer to the DB2 Information Center.

Investigate the amount of memory that was used for the database heap over time to determine the most appropriate value for the database heap configuration parameter. The database system monitor tracks the highest amount of memory that was used for the database heap.

Take one of the following actions:

Launch DB2 tool: Memory Visualizer

The Memory Visualizer is used to monitor memory allocation within a DB2 instance. It can be used to monitor overall memory usage, and to update configuration parameters for individual memory components.

To open the Memory Visualizer:

1. From the Control Center, expand the object tree until you find the instances folder.
2. Click the instances folder. Any existing instances are displayed in the contents pane on the right side of the window.
3. Right-click the instance that you want in the contents pane, and click View Memory Usage in the pop-up menu. The Memory Visualizer opens.

To start the Memory Visualizer from the command line issue the db2memvis command.

The Memory Visualizer displays a hierarchical list of memory pools for the database manager. Database Heap is listed under the Database Manager Memory group for each database. On Windows, it is listed under the Database Manager Shared Memory group.

Click the check box on the Show Plot column for the Database Heap row to add the element to the plot.

### Usage notes:

The GET RECOMMENDATIONS FOR HEALTH INDICATOR command can be used in two different ways:

- Specify only the health indicator to get an informational list of all possible recommendations. If no object is specified, the command will return a full listing of all recommendations that can be used to resolve an alert on the given health indicator.
- Specify an object to resolve a specific alert on that object. If an object (for example, a database or a table space) is specified, the recommendations returned will be specific to an alert on the object identified. In this case, the recommendations will be more specific and will contain more information about resolving the alert. If the health indicator identified is not in an alert state on the specified object, no recommendations will be returned.

### Related tasks:

## GET RECOMMENDATIONS FOR HEALTH INDICATOR

- “Retrieving health recommendations using a client application” in *System Monitor Guide and Reference*

### **Related reference:**

- “db2GetRecommendations API - Get recommendations to resolve a health indicator in alert state” in *Administrative API Reference*
- “AUTOCONFIGURE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “Health indicators” in *System Monitor Guide and Reference*
- “Health indicators summary” in *System Monitor Guide and Reference*

## GET ROUTINE

Retrieves a routine SQL Archive (SAR) file for a specified SQL routine.

### Authorization:

*dbadm*

### Required connection:

Database. If implicit connect is enabled, a connection to the default database is established.

### Command syntax:

```

▶▶ GET ROUTINE INTO file_name FROM SPECIFIC PROCEDURE routine_name
HIDE BODY

```

### Command parameters:

#### INTO *file\_name*

Names the file where routine SQL archive (SAR) is stored.

#### FROM

Indicates the start of the specification of the routine to be retrieved.

#### SPECIFIC

The specified routine name is given as a specific name.

#### PROCEDURE

The routine is an SQL procedure.

#### *routine\_name*

The name of the procedure. If SPECIFIC is specified then it is the specific name of the procedure. If the name is not qualified with a schema name, the CURRENT SCHEMA is used as the schema name of the routine. The *routine-name* must be an existing procedure that is defined as an SQL procedure.

#### HIDE BODY

Specifies that the body of the routine must be replaced by an empty body when the routine text is extracted from the catalogs.

This does not affect the compiled code; it only affects the text.

### Examples:

```
GET ROUTINE INTO procs/proc1.sar FROM PROCEDURE myappl.proc1;
```

### Usage Notes:

If a GET ROUTINE or a PUT ROUTINE operation (or their corresponding procedure) fails to execute successfully, it will always return an error (SQLSTATE 38000), along with diagnostic text providing information about the cause of the failure. For example, if the procedure name provided to GET ROUTINE does not identify an SQL procedure, diagnostic "-204, 42704" text will be returned, where "-204" and "42704" are the SQLCODE and SQLSTATE, respectively, that identify the

## GET ROUTINE

cause of the problem. The SQLCODE and SQLSTATE in this example indicate that the procedure name provided in the GET ROUTINE command is undefined.

**Related reference:**

- “PUT ROUTINE ” on page 609
- “PUT\_ROUTINE\_SAR procedure” in *Administrative SQL Routines and Views*



---

## GET SNAPSHOT

Collects status information and formats the output for the user. The information returned represents a *snapshot* of the database manager operational status at the time the command was issued.

**Scope:**

In a partitioned database environment, this command can be invoked from any database partition defined in the `db2nodes.cfg` file. It acts only on that database partition.

**Authorization:**

One of the following:

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

**Required connection:**

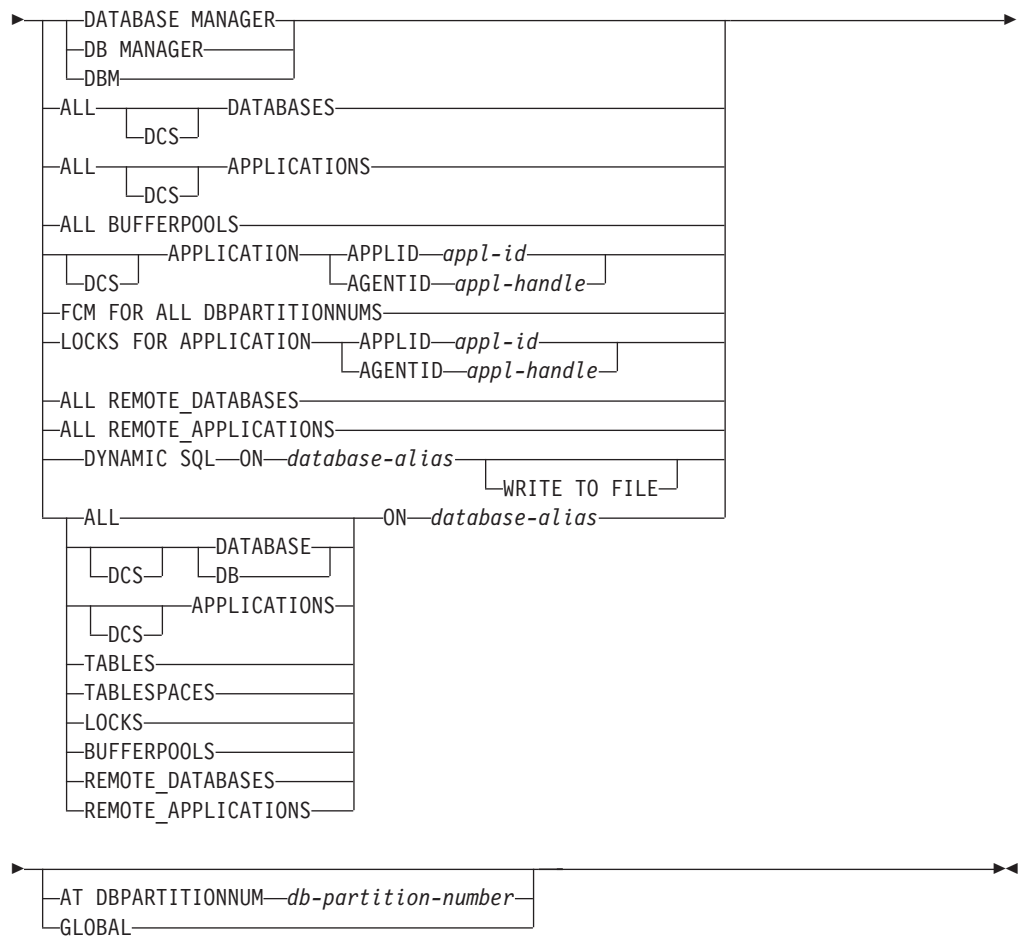
Instance. If there is no instance attachment, a default instance attachment is created.

To obtain a snapshot of a remote instance, it is necessary to first attach to that instance.

**Command syntax:**

►►—GET SNAPSHOT FOR—————▶

## GET SNAPSHOT



The monitor switches must be turned on in order to collect some statistics.

### Command parameters:

#### DATABASE MANAGER

Provides statistics for the active database manager instance.

#### ALL DATABASES

Provides general statistics for all active databases on the current database partition.

#### ALL APPLICATIONS

Provides information about all active applications that are connected to a database on the current database partition.

#### 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 the LIST APPLICATIONS command.

#### 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 the LIST APPLICATIONS command to get a specific application handle.

**FCM FOR ALL DBPARTITIONNUMS**

Provides Fast Communication Manager (FCM) statistics between the database partition against which the GET SNAPSHOT command was issued and the other database partitions in the partitioned database environment.

**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 REMOTE\_DATABASES**

Provides general statistics about all active remote databases on the current database partition.

**ALL REMOTE\_APPLICATIONS**

Provides information about all active remote applications that are connected to the current database partition.

**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.

**REMOTE\_DATABASES ON *database-alias***

Provides general statistics about all active remote databases for a specified database.

**REMOTE\_APPLICATIONS ON *database-alias***

Provides information about remote applications for a 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

## GET SNAPSHOT

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.

**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.

**AT DBPARTITIONNUM db-partition-number**

Returns results for the database partition specified.

**GLOBAL**

Returns an aggregate result for all database partitions in a partitioned database system.

### Examples:

- To request snapshot information about the database manager, issue:  
`get snapshot for database manager`
- To request snapshot information about the SAMPLE database:  
`get snapshot for database on sample`
- To request snapshot information about a specific application with application handle 765 connected to the SAMPLE database, issue:  
`get snapshot for application agentid 765`
- To request dynamic SQL snapshot information about the SAMPLE database, issue:  
`get snapshot for dynamic sql on sample`

### Usage notes:

- When write suspend is ON against a database, snapshots cannot be issued against that database until write suspend is turned OFF. When a snapshot is issued against a database for which write suspend was turned on, a diagnostic probe is written to the db2diag.log and that database is skipped.
- 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. If the recording switch TIMESTAMP has been set to off, timestamp related elements will report "Not Collected".
- 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.

However, if a REORG TABLE is being performed or has been performed during this period, some information is returned although some fields are not displayed.

- To obtain snapshot information from all database partitions (which is different than the aggregate result of all partitions), the snapshot administrative views should be used.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword `NODE` can be substituted for `DBPARTITIONNUM`.
- The keyword `NODES` can be substituted for `DBPARTITIONNUMS`.

### Related concepts:

- “Snapshot monitor” in *System Monitor Guide and Reference*

### Related tasks:

- “Capturing a database snapshot from a client application” in *System Monitor Guide and Reference*
- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

### Related reference:

- “db2GetSnapshot API - Get a snapshot of the database manager operational status” in *Administrative API Reference*
- “GET MONITOR SWITCHES ” on page 478
- “LIST APPLICATIONS ” on page 520
- “RESET MONITOR ” on page 671
- “UPDATE MONITOR SWITCHES ” on page 782

---

**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 \sql11ib\doc\html subdirectory.

The command line processor will not know if the command succeeds or fails, and cannot report error conditions.

**Related tasks:**

- "Invoking command help from the command line processor" on page 326

---

## HISTORY

Displays the history of commands run within a CLP interactive mode session.

### Scope

This command can only be run within CLP interactive mode. Specifically, it cannot be run from the CLP command mode or the CLP batch mode.

### Authorization:

None

### Required connection:

None

### Command syntax:



### Command parameters:

#### REVERSE

Displays the command history in reverse order, with the most-recently run command listed first. If this parameter is not specified, the commands are listed in chronological order, with the most recently run command listed last.

*num* Displays only the most recent *num* commands. If this parameter is not specified, a maximum of 20 commands are displayed. However, the number of commands that are displayed is also restricted by the number of commands that are stored in the command history.

### Usage notes:

1. The value of the DB2\_CLP\_HISTSIZ variable specifies the maximum number of commands to be stored in the command history. This registry variable can be set to any value between 1 and 500 inclusive. If this registry variable is not set or is set to a value outside the valid range, a maximum of 20 commands is stored in the command history.
2. Since the HISTORY command will always be listed in the command history, the maximum number of commands displayed will always be one greater than the user-specified maximum.
3. The command history is not persistent across CLP interactive mode sessions, which means that the command history is not saved at the end of an interactive mode session.
4. The command histories of multiple concurrently running CLP interactive mode sessions are independent of one another.

### Related reference:

- “EDIT ” on page 432
- “RUNCMD ” on page 701

---

## IMPORT

Inserts data from an external file with a supported file format into a table, hierarchy, view or nickname. LOAD is a faster alternative, but 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, view, or nickname
  - INSERT and SELECT privilege on each participating table or view
- IMPORT to an existing table using the INSERT\_UPDATE option, requires one of the following:
  - *sysadm*
  - *dbadm*
  - CONTROL privilege on each participating table, view, or nickname
  - INSERT, SELECT, UPDATE and DELETE privilege on each participating table or view
- IMPORT to an existing table using the REPLACE or REPLACE\_CREATE option, requires one of the following:
  - *sysadm*
  - *dbadm*
  - CONTROL privilege on the table or view
  - INSERT, SELECT, and DELETE privilege on the table or view
- IMPORT to a new table using the CREATE or REPLACE\_CREATE option, requires one of the following:
  - *sysadm*
  - *dbadm*
  - CREATETAB authority on the database and USE privilege on the table space, as well as one of:
    - IMPLICIT\_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
    - CREATIN privilege on the schema, if the schema name of the table refers to an existing schema
- IMPORT to 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 USE privilege on the table space 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
- To import data into a table that has protected columns, the session authorization ID must have LBAC credentials that allow write access to all protected columns in the table. Otherwise the import fails and an error (SQLSTATE 42512) is returned.
- To import data into a table that has protected rows, the session authorization ID must hold LBAC credentials that meets these criteria:
  - It is part of the security policy protecting the table
  - It was granted to the session authorization ID for write access

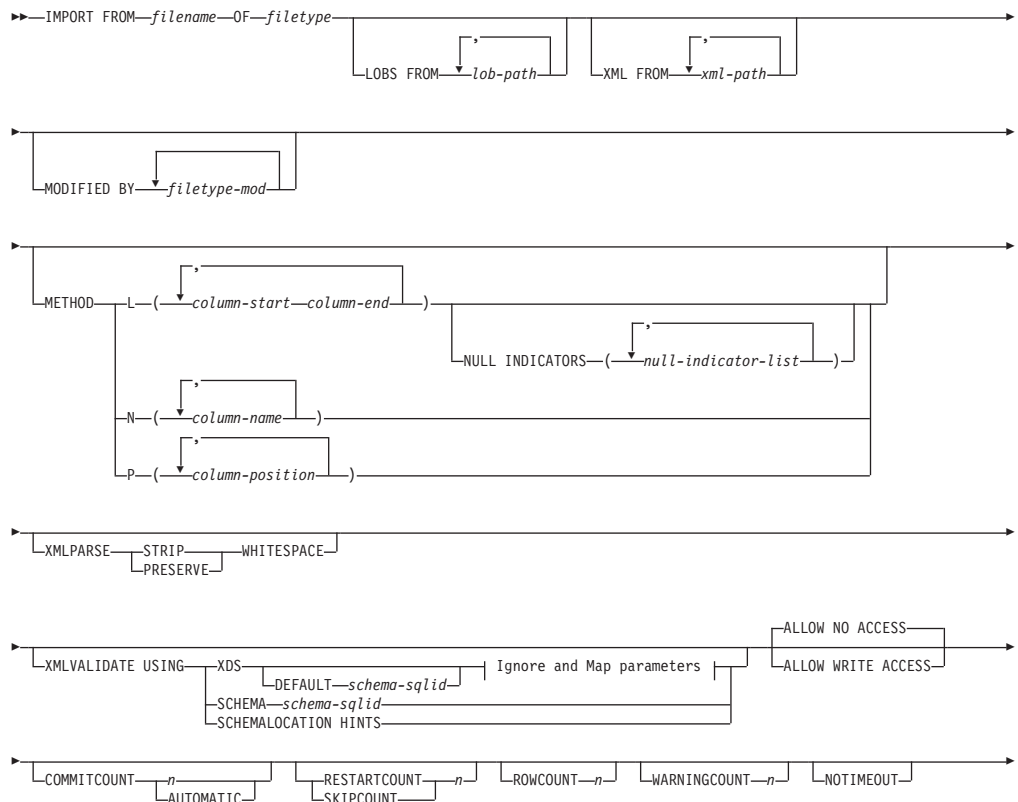
The label on the row to insert, the user’s LBAC credentials, the security policy definition, and the LBAC rules determine the label on the row.

- If the REPLACE or REPLACE\_CREATE option is specified, the session authorization ID must have the authority to drop the table.

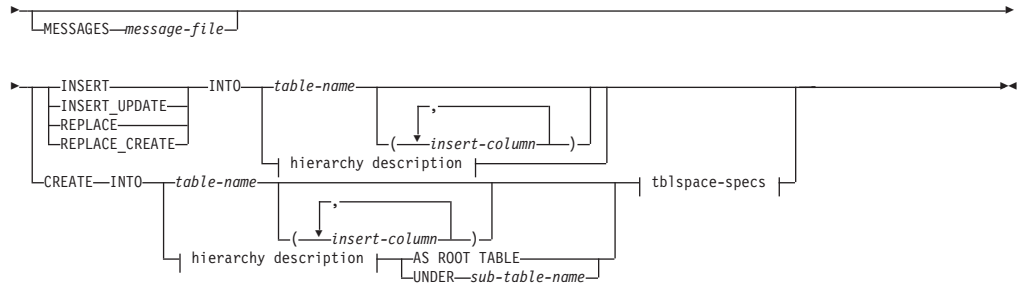
**Required connection:**

Database. If implicit connect is enabled, a connection to the default database is established. Utility access to Linux, UNIX, or Windows database servers from Linux, UNIX, or Windows clients must be a direct connection through the engine and not through a DB2 Connect gateway or loop back environment.

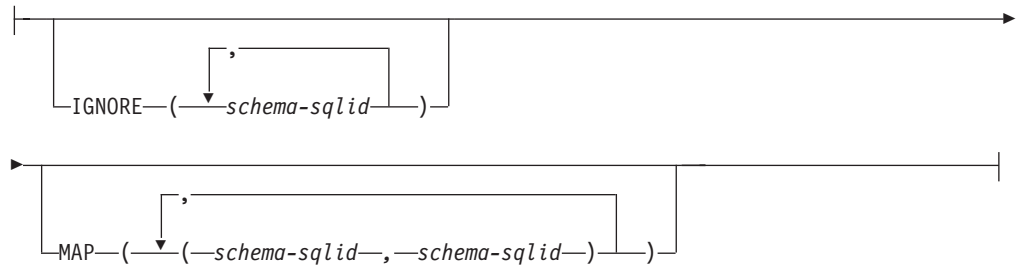
**Command syntax:**



# IMPORT



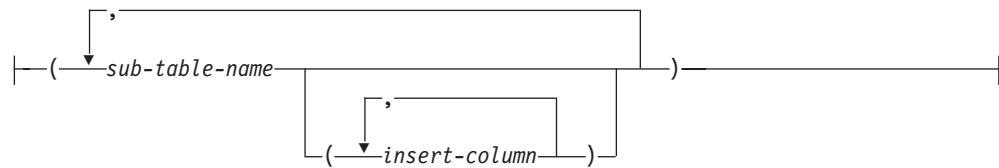
## Ignore and Map parameters:



## hierarchy description:



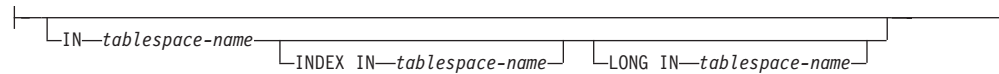
## sub-table-list:



## traversal-order-list:



## tblspace-specs:



## 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.

**ALLOW NO ACCESS**

Runs import in the offline mode. An exclusive (X) lock on the target table is acquired before any rows are inserted. This prevents concurrent applications from accessing table data. This is the default import behavior.

**ALLOW WRITE ACCESS**

Runs import in the online mode. An intent exclusive (IX) lock on the target table is acquired when the first row is inserted. This allows concurrent readers and writers to access table data. Online mode is not compatible with the REPLACE, CREATE, or REPLACE\_CREATE import options. Online mode is not supported in conjunction with buffered inserts. The import operation will periodically commit inserted data to prevent lock escalation to a table lock and to avoid running out of active log space. These commits will be performed even if the COMMITCOUNT option was not used. During each commit, import will lose its IX table lock, and will attempt to reacquire it after the commit. This parameter is required when you import to a nickname and COMMITCOUNT must be specified with a valid number (AUTOMATIC is not considered a valid option).

**AS ROOT TABLE**

Creates one or more sub-tables as a stand-alone table hierarchy.

**COMMITCOUNT *n*/AUTOMATIC**

Performs a COMMIT after every *n* records are imported. When a number *n* is specified, import performs a COMMIT after every *n* records are imported. When compound inserts are used, a user-specified commit frequency of *n* is rounded up to the first integer multiple of the compound count value. When AUTOMATIC is specified, import internally determines when a commit needs to be performed. The utility will commit for either one of two reasons:

- to avoid running out of active log space
- to avoid lock escalation from row level to table level

If the ALLOW WRITE ACCESS option is specified, and the COMMITCOUNT option is not specified, the import utility will perform commits as if COMMITCOUNT AUTOMATIC had been specified.

If the **IMPORT** command encounters an SQL0964C (Transaction Log Full) while inserting or updating a record and the COMMITCOUNT *n* is specified, **IMPORT** will attempt to resolve the issue by performing an unconditional commit and then reattempt to insert or update the record. If this does not help resolve the log full condition (which would be the case when the log full is attributed to other activity on the database), then the **IMPORT** command will fail as expected, however the number of rows committed may not be a multiple of the COMMITCOUNT *n* value. The RESTARTCOUNT or SKIPCOUNT option can be used to avoid processing those row already committed.

**CREATE**

Creates the table definition and row contents in the code page of the database. 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.

This parameter is not valid when you import to a nickname.

**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

## IMPORT

than 254, CREATE might fail because the rows are too long. See Using import to recreate an exported table for a list of restrictions. In this case, the table should be created manually, and IMPORT with INSERT should be invoked, or, alternatively, the LOAD command should be used.

### **DEFAULT schema-sqlid**

This option can only be used when the USING XDS parameter is specified. The schema specified through the DEFAULT clause identifies a schema to use for validation when the XML Data Specifier (XDS) of an imported XML document does not contain an SCH attribute identifying an XML Schema.

The DEFAULT clause takes precedence over the IGNORE and MAP clauses. If an XDS satisfies the DEFAULT clause, the IGNORE and MAP specifications will be ignored.

### **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.

### **IGNORE schema-sqlid**

This option can only be used when the USING XDS parameter is specified. The IGNORE clause specifies a list of one or more schemas to ignore if they are identified by an SCH attribute. If an SCH attribute exists in the XML Data Specifier for an imported XML document, and the schema identified by the SCH attribute is included in the list of schemas to IGNORE, then no schema validation will occur for the imported XML document.

If a schema is specified in the IGNORE clause, it cannot also be present in the left side of a schema pair in the MAP clause.

The IGNORE clause applies only to the XDS. A schema that is mapped by the MAP clause will not be subsequently ignored if specified by the IGNORE clause.

### **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 or LARGE 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, a declared temporary 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. The maximum number of paths that can be specified is 999. This will implicitly activate the LOBSINFILFILE behaviour.

This parameter is not valid when you import to a nickname.

**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 LARGE DMS table space.

**MAP schema-sqlid**

This option can only be used when the USING XDS parameter is specified. Use the MAP clause to specify alternate schemas to use in place of those specified by the SCH attribute of an XML Data Specifier (XDS) for each imported XML document. The MAP clause specifies a list of one or more schema pairs, where each pair represents a mapping of one schema to another. The first schema in the pair represents a schema that is referred to by an SCH attribute in an XDS. The second schema in the pair represents the schema that should be used to perform schema validation.

If a schema is present in the left side of a schema pair in the MAP clause, it cannot also be specified in the IGNORE clause.

Once a schema pair mapping is applied, the result is final. The mapping operation is non-transitive, and therefore the schema chosen will not be subsequently applied to another schema pair mapping.

A schema cannot be mapped more than once, meaning that it cannot appear on the left side of more than one pair.

**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.

## IMPORT

### METHOD

**L** Specifies the start and end column numbers from which to import data. A column number is a byte offset from the beginning of a row of data. It is numbered starting from 1.

**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 field numbers of the input data fields 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 file type modifier options. See File type modifiers for the import utility.

### NOTIMEOUT

Specifies that the import utility will not time out while waiting for locks. This option supersedes the *locktimeout* database configuration parameter. Other applications are not affected.

### NULL INDICATORS null-indicator-list

This option can only be used when the METHOD L parameter is specified. That is, the input file is an ASC file. The null indicator list is a comma-separated list of positive integers specifying the column number of each null indicator field. The column number is the byte offset of the null indicator field from the beginning of a row of data. There must be one entry in the null indicator list for each data field defined in the METHOD L parameter. A column number of zero indicates that the corresponding data field always contains data.

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 imported.

The NULL indicator character can be changed using the MODIFIED BY option, with the nullindchar file type modifier.

### 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.

The WSF file type is not supported when you import to a nickname.

**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. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

This parameter is not valid when you import to a nickname.

This option does not honour the CREATE TABLE statement's NOT LOGGED INITIALLY (NLI) clause or the ALTER TABLE statement's ACTIVE NOT LOGGED INITIALLY clause.

If an import with the REPLACE option is performed within the same transaction as a CREATE TABLE or ALTER TABLE statement where the NLI clause is invoked, the import will not honor the NLI clause. All inserts will be logged.

**Workaround 1**

Delete the contents of the table using the DELETE statement, then invoke the import with INSERT statement

**Workaround 2**

Drop the table and recreate it, then invoke the import with INSERT statement.

This limitation applies to DB2 UDB Version 7 and DB2 UDB Version 8

**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, in the code page of the database. See Using import to recreate an exported table for a list of restrictions.

This option can only be used with IXF files. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

This parameter is not valid when you import to a nickname.

**RESTARTCOUNT *n***

Specifies that an import operation is to be started at record  $n + 1$ . The first  $n$  records are skipped. This option is functionally equivalent to SKIPCOUNT. RESTARTCOUNT and SKIPCOUNT are mutually exclusive.

**ROWCOUNT *n***

Specifies the number  $n$  of physical records in the file to be imported (inserted or updated). Allows a user to import only  $n$  rows from a file, starting from the record determined by the SKIPCOUNT or RESTARTCOUNT options. If the SKIPCOUNT or RESTARTCOUNT options are not specified, the first  $n$  rows are imported. If SKIPCOUNT  $m$  or RESTARTCOUNT  $m$  is specified, rows  $m+1$  to  $m+n$  are imported. When compound inserts are used, user specified rowcount  $n$  is rounded up to the first integer multiple of the compound count value.

**SKIPCOUNT *n***

Specifies that an import operation is to be started at record  $n + 1$ . The first

## IMPORT

*n* records are skipped. This option is functionally equivalent to RESTARTCOUNT. SKIPCOUNT and RESTARTCOUNT are mutually exclusive.

### **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.

### **WARNINGCOUNT *n***

Stops the import 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 the import file or the target table is specified incorrectly, the import utility will generate a warning for each row that it attempts to import, which will cause the import to fail. If *n* is zero, or this option is not specified, the import operation will continue regardless of the number of warnings issued.

### **XML FROM xml-path**

Specifies one or more paths that contain the XML files.

### **XMLPARSE**

Specifies how XML documents are parsed. If this option is not specified, the parsing behaviour for XML documents will be determined by the value of the CURRENT XMLPARSE OPTION special register.

### **STRIP WHITESPACE**

Specifies to remove whitespace when the XML document is parsed.

### **PRESERVE WHITESPACE**

Specifies not to remove whitespace when the XML document is parsed.

### **XMLVALIDATE**

Specifies that XML documents are validated against a schema, when applicable.

### **USING XDS**

XML documents are validated against the XML schema identified by the XML Data Specifier (XDS) in the main data file. By default, if the XMLVALIDATE option is invoked with the USING XDS clause, the schema used to perform validation will be determined by the SCH attribute of the XDS. If an SCH attribute is not present in the XDS, no schema validation will occur unless a default schema is specified by the DEFAULT clause.

The DEFAULT, IGNORE, and MAP clauses can be used to modify the schema determination behavior. These three optional clauses



apply directly to the specifications of the XDS, and not to each other. For example, if a schema is selected because it is specified by the DEFAULT clause, it will not be ignored if also specified by the IGNORE clause. Similarly, if a schema is selected because it is specified as the first part of a pair in the MAP clause, it will not be re-mapped if also specified in the second part of another MAP clause pair.

#### **USING SCHEMA schema-sqlid**

XML documents are validated against the XML schema with the specified SQL identifier. In this case, the SCH attribute of the XML Data Specifier (XDS) will be ignored for all XML columns.

#### **USING SCHEMALOCATION HINTS**

XML documents are validated against the schemas identified by XML schema location hints in the source XML documents. If a schemaLocation attribute is not found in the XML document, no validation will occur. When the USING SCHEMALOCATION HINTS clause is specified, the SCH attribute of the XML Data Specifier (XDS) will be ignored for all XML columns.

See examples of the XMLVALIDATE option below.

#### **Examples:**

##### **Example 1**

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.
```

##### **Example 2 (Importing into a Table with an Identity Column)**

TABLE1 has 4 columns:

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 is the same as TABLE1, except that C2 is a GENERATED ALWAYS identity column.

## IMPORT

Data records in DATAFILE1 (DEL format):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

Data records in DATAFILE2 (DEL format):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

The following command generates identity values for rows 1 and 2, since no identity values are supplied in DATAFILE1 for those rows. Rows 3 and 4, however, are assigned the user-supplied identity values of 100 and 101, respectively.

```
db2 import from datafile1.del of del replace into table1
```

To import DATAFILE1 into TABLE1 so that identity values are generated for all rows, issue one of the following commands:

```
db2 import from datafile1.del of del method P(1, 3, 4)
 replace into table1 (c1, c3, c4)
db2 import from datafile1.del of del modified by identityignore
 replace into table1
```

To import DATAFILE2 into TABLE1 so that identity values are generated for each row, issue one of the following commands:

```
db2 import from datafile2.del of del replace into table1 (c1, c3, c4)
db2 import from datafile2.del of del modified by identitymissing
 replace into table1
```

If DATAFILE1 is imported into TABLE2 without using any of the identity-related file type modifiers, rows 1 and 2 will be inserted, but rows 3 and 4 will be rejected, because they supply their own non-NULL values, and the identity column is GENERATED ALWAYS.

### Examples of using the XMLVALIDATE clause:

#### Example 1 (XMLVALIDATE USING XDS)

For the following XMLVALIDATE clause:

```
XMLVALIDATE USING XDS
 IGNORE (S1.SCHEMA_A)
 MAP ((S1.SCHEMA_A, S2.SCHEMA_B))
```

The import would fail due to invalid syntax, since the IGNORE of S1.SCHEMA\_A would conflict with the MAP of S1.SCHEMA\_A to S2.SCHEMA\_B.

#### Example 2 (XMLVALIDATE USING XDS)

For the following XMLVALIDATE clause:

```
XMLVALIDATE USING XDS
 DEFAULT S8.SCHEMA_H
 IGNORE (S9.SCHEMA_I, S10.SCHEMA_J)
 MAP ((S1.SCHEMA_A, S2.SCHEMA_B), (S3.SCHEMA_C, S5.SCHEMA_E),
 (S6.SCHEMA_F, S3.SCHEMA_C), (S4.SCHEMA_D, S7.SCHEMA_G))
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema with SQL identifier "S8.SCHEMA\_H" is used to validate the document in file "xmlfile.001.xml", since "S8.SCHEMA\_H" was specified as the default schema to use.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' OFF='10' LEN='500' SCH='S10.SCHEMA_J' />
```

No schema validation occurs for the document in file "xmlfile.002.xml", since although the XDS specifies "S10.SCHEMA\_J" as the schema to use, that schema is part of the IGNORE clause. The document contents can be found at byte offset 10 in the file (meaning the 11th byte), and is 500 bytes long.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.003.xml' SCH='S6.SCHEMA_F' />
```

The XML schema with SQL identifier "S3.SCHEMA\_C" is used to validate the document in file "xmlfile.003.xml". This is because the MAP clause specifies that schema "S6.SCHEMA\_F" should be mapped to schema "S3.SCHEMA\_C". Note that further mapping does not take place, therefore the mapping of schema "S3.SCHEMA\_C" to schema "S5.SCHEMA\_E" does not apply in this case.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.004.xml' SCH='S11.SCHEMA_K' />
```

The XML schema with SQL identifier "S11.SCHEMA\_K" is used to validate the document in file "xmlfile.004.xml". Note that none of the DEFAULT, IGNORE, or MAP specifications apply in this case.

### Example 3 (XMLVALIDATE USING XDS)

For the following XMLVALIDATE clause:

```
XMLVALIDATE USING XDS
 DEFAULT S1.SCHEMA_A
 IGNORE (S1.SCHEMA_A)
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema with SQL identifier "S1.SCHEMA\_A" is used to validate the document in file "xmlfile.001.xml", since "S1.SCHEMA\_1" was specified as the default schema to use.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

No schema validation occurs for the document in file "xmlfile.002", since although the XDS specifies "S1.SCHEMA\_A" as the schema to use, that schema is part of the IGNORE clause.

### Example 4 (XMLVALIDATE USING XDS)

For the following XMLVALIDATE clause:

## IMPORT

```
XMLVALIDATE USING XDS
 DEFAULT S1.SCHEMA_A
 MAP ((S1.SCHEMA_A, S2.SCHEMA_B), (S2.SCHEMA_B, S1.SCHEMA_A))
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema with SQL identifier "S1.SCHEMA\_A" is used to validate the document in file "xmlfile.001.xml", since "S1.SCHEMA\_1" was specified as the default schema to use. Note that since the DEFAULT clause was applied, the MAP clause is not subsequently applied. Therefore the mapping of schema "S1.SCHEMA\_A" to schema "S2.SCHEMA\_B" does not apply in this case.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The XML schema with SQL identifier "S2.SCHEMA\_B" is used to validate the document in file "xmlfile.002.xml". This is because the MAP clause specifies that schema "S1.SCHEMA\_A" should be mapped to schema "S2.SCHEMA\_B". Note that further mapping does not take place, therefore the mapping of schema "S2.SCHEMA\_B" to schema "S1.SCHEMA\_A" does not apply in this case.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.003.xml' SCH='S2.SCHEMA_B' />
```

The XML schema with SQL identifier "S1.SCHEMA\_A" is used to validate the document in file "xmlfile.003.xml". This is because the MAP clause specifies that schema "S2.SCHEMA\_B" should be mapped to schema "S1.SCHEMA\_A". Note that further mapping does not take place, therefore the mapping of schema "S1.SCHEMA\_A" to schema "S2.SCHEMA\_B" does not apply in this case.

### Example 5 (XMLVALIDATE USING SCHEMA)

For the following XMLVALIDATE clause:

```
XMLVALIDATE USING SCHEMA S2.SCHEMA_B
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The document in file

```
xmlfile.001.xml
```

is validated using the XML schema with SQL identifier "S2.SCHEMA\_B".

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The document in file "xmlfile.002.xml" is validated using the XML schema with SQL identifier "S2.SCHEMA\_B". Note that the SCH attribute is ignored, since validation is being performed using a schema specified by the USING SCHEMA clause.

### Example 6 (XMLVALIDATE USING USING SCHEMALOCATION HINTS)

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema used is determined by the schemaLocation attribute in the document contents, and no validation would occur if one is not present.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The XML schema used is determined by the schemaLocation attribute in the document contents, and no validation would occur if one is not present. Note that the SCH attribute is ignored, since validation is being performed using SCHEMALOCATION HINTS.

### 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. If automatic COMMITs are not performed, a full log results in a ROLLBACK.

Offline import does not perform automatic COMMITs if any of the following conditions is true:

- the target is a view, not a table
- compound inserts are used
- buffered inserts are used

## IMPORT

By default, online import performs automatic COMMITs to free both the active log space and the lock list. Automatic COMMITs are not performed only if a COMMITCOUNT value of zero is specified.

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 might 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, fields containing the row separators will shrink or expand. 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 clients on the AIX operating system.

For table objects on an 8 KB page that are close to the limit of 1012 columns, import of PC/IXF data files might 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 use **db2look** to dump the DDL statement that created the table, 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, a declared temporary table, or a summary table.

Views cannot be created through the import utility.

On the Windows operating system:

- Importing logically split PC/IXF files is not supported.
- Importing bad format PC/IXF or WSF files is not supported.

Security labels in their internal format might contain newline characters. If you import the file using the DEL file format, those newline characters can be mistaken for delimiters. If you have this problem use the older default priority for delimiters by specifying the `delprioritychar` file type modifier in the IMPORT command.

#### **Federated considerations:**

When using the IMPORT command and the INSERT, UPDATE, or INSERT\_UPDATE command parameters, you must ensure that you have

## IMPORT

CONTROL privilege on the participating nickname. You must ensure that the nickname you wish to use when doing an import operation already exists. There are also several restrictions you should be aware of as shown in the IMPORT command parameters section.

### Related concepts:

- “Import Overview” in *Data Movement Utilities Guide and Reference*
- “Privileges, authorities, and authorization required to use import” in *Data Movement Utilities Guide and Reference*

### Related tasks:

- “Importing data” in *Data Movement Utilities Guide and Reference*

### Related reference:

- “XMLPARSE scalar function” in *SQL Reference, Volume 1*
- “ADMIN\_CMD procedure – Run administrative commands” in *Administrative SQL Routines and Views*
- “db2look - DB2 statistics and DDL extraction tool ” on page 146
- “IMPORT command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “Import sessions - CLP examples” in *Data Movement Utilities Guide and Reference*
- “LOB and XML file behavior with regard to import and export” in *Data Movement Utilities Guide and Reference*



## INITIALIZE TAPE

Initializes tapes for backup and restore operations to streaming tape devices. This command is only supported on Windows operating systems.

### Authorization:

One of the following:

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

### 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.

The buffer size specified for the BACKUP DATABASE command and for RESTORE DATABASE 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.

When backing up to tape, use of a variable block size is currently not supported. If you must use this option, ensure that you have well tested procedures in place that enable you to recover successfully, using backup images that were created with a variable block size.

When using a variable block size, you must specify a backup buffer size that is less than or equal to the maximum limit for the tape devices that you are using. For optimal performance, the buffer size must be equal to the maximum block size limit of the device being used.

### Related reference:

- “BACKUP DATABASE ” on page 349
- “RESTORE DATABASE ” on page 675
- “REWIND TAPE ” on page 690
- “SET TAPE POSITION ” on page 723
- “INITIALIZE TAPE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

## INSPECT

Inspect database for architectural integrity, checking the pages of the database for page consistency. The inspection checks the structures of table objects and structures of table spaces are valid.

### Scope:

In a single partition database environment, the scope is that single partition only. In a partitioned database system, it is the collection of all logical partitions defined in db2nodes.cfg. For partitioned tables, the CHECK DATABASE and CHECK TABLESPACE options include individual data partitions and non-partitioned indexes. The CHECK TABLE option is also available for a partitioned table, however it will check all data partitions and indexes in a table, rather than checking a single data partition or index.

### Authorization:

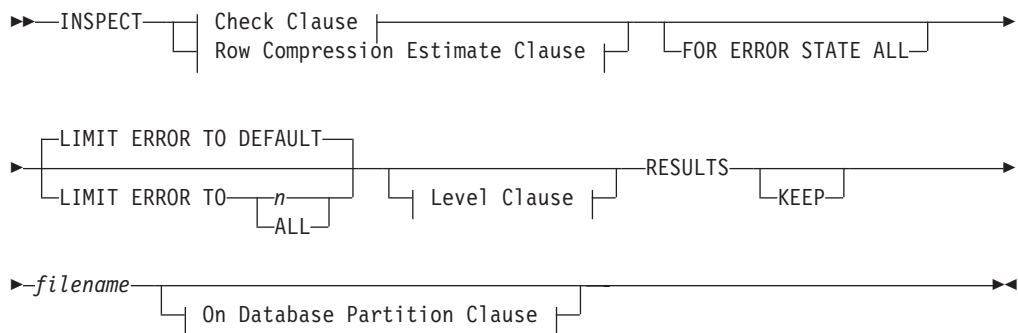
For INSPECT CHECK, one of the following:

- *sysadm*
- *dbadm*
- *sysctrl*
- *sysmaint*
- CONTROL privilege if single table.

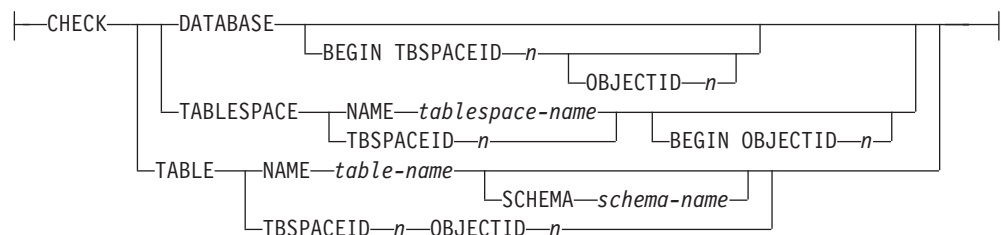
### Required Connection:

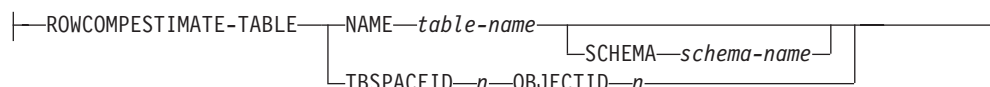
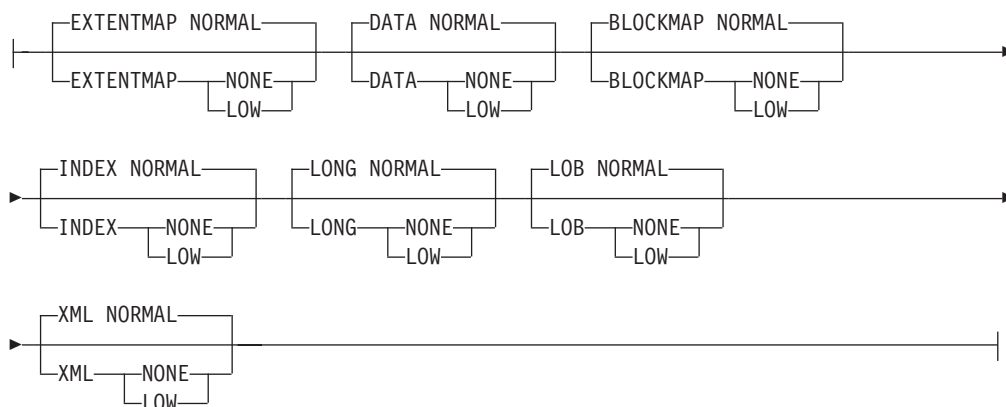
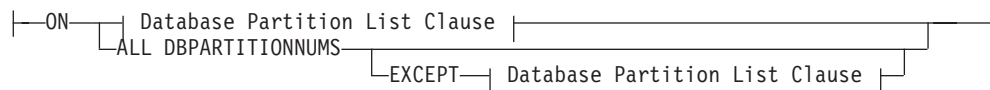
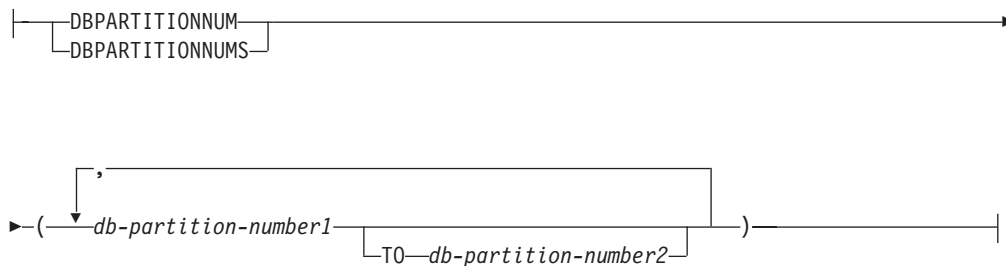
Database

### Command Syntax:



### Check Clause:



**Row Compression Estimate Clause:****Level Clause:****On Database Partition Clause:****Database Partition List Clause:****Command Parameters:****CHECK**

Specifies check processing.

**DATABASE**

Specifies whole database.

**BEGIN TBSpaceID n**

Specifies processing to begin from table space with given table space ID number.

**BEGIN TBSpaceID n OBJECTID n**

Specifies processing to begin from table with given table space ID number and object ID number.

**TABLESPACE**

## INSPECT

### NAME **tablespace-name**

Specifies single table space with given table space name.

### TBSPACEID **n**

Specifies single table space with given table space ID number.

### BEGIN OBJECTID **n**

Specifies processing to begin from table with given object ID number.

## TABLE

### NAME **table-name**

Specifies table with given table name.

### SCHEMA **schema-name**

Specifies schema name for specified table name for single table operation.

### TBSPACEID **n** OBJECTID **n**

Specifies table with given table space ID number and object ID number.

## ROWCOMPESTIMATE

Estimates the effectiveness of row compression for a table. You can also specify which database partition(s) this operation is to be done on.

This tool is capable of taking a sample of the table data, and building a dictionary from it. This dictionary can then be used to test compression against the records contained in the sample. From this test compression, data is gathered from which the following estimates are made:

- Percentage of bytes saved from compression
- Percentage of pages saved from compression
- Percentage rows ineligible for compression due to small data size
- Compression dictionary size
- Expansion dictionary size

**INSPECT** will insert the dictionary built for gathering these compression estimates if the **COMPRESS YES** attribute is set for this table, and a dictionary does not already exist for this table. **INSPECT** will attempt to insert the dictionary concurrent to other applications accessing the table. Dictionary insert requires an Exclusive Table Alter lock and an Intent on Exclusive Table lock. **INSPECT** will only insert a dictionary into tables that support Row Compression. For partitioned tables, a separate dictionary is built and inserted on each partition.

## RESULTS

Specifies the result output file. The file will be written out to the diagnostic data directory path. If there is no error found by the check processing, this result output file will be erased at the end of the **INSPECT** operation. If there are errors found by the check processing, this result output file will not be erased at the end of the **INSPECT** operation.

**KEEP** Specifies to always keep the result output file.

### **file-name**

Specifies the name for the result output file.

## ALL DBPARTITIONNUMS

Specifies that operation is to be done on all database partitions specified in the **db2nodes.cfg** file. This is the default if a node clause is not specified.

**EXCEPT**

Specifies that operation is to be done on all database partitions specified in the db2nodes.cfg file, except those specified in the node list.

**ON DBPARTITIONNUM / ON DBPARTITIONNUMS**

Perform operation on a set of database partitions.

**db-partition-number1**

Specifies a database partition number in the database partition list.

**db-partition-number2**

Specifies the second database partition number, so that all database partitions from db-partition-number1 up to and including db-partition-number2 are included in the database partition list.

**FOR ERROR STATE ALL**

For table object with internal state already indicating error state, the check will just report this status and not scan through the object. Specifying this option will have the processing scan through the object even if internal state already lists error state.

**LIMIT ERROR TO *n***

Number of pages in error for an object to limit reporting for. When this limit of the number of pages in error for an object is reached, the processing will discontinue the check on the rest of the object.

**LIMIT ERROR TO DEFAULT**

Default number of pages in error for an object to limit reporting for. This value is the extent size of the object. This parameter is the default.

**LIMIT ERROR TO ALL**

No limit on number of pages in error reported.

**EXTENTMAP****NORMAL**

Specifies processing level is normal for extent map. Default.

**NONE**

Specifies processing level is none for extent map.

**LOW**

Specifies processing level is low for extent map.

**DATA****NORMAL**

Specifies processing level is normal for data object. Default.

**NONE**

Specifies processing level is none for data object.

**LOW**

Specifies processing level is low for data object.

**BLOCKMAP****NORMAL**

Specifies processing level is normal for block map object. Default.

**NONE**

Specifies processing level is none for block map object.

**LOW**

Specifies processing level is low for block map object.

**INDEX**

## INSPECT

### NORMAL

Specifies processing level is normal for index object. Default.

### NONE

Specifies processing level is none for index object.

**LOW** Specifies processing level is low for index object.

## LONG

### NORMAL

Specifies processing level is normal for long object. Default.

### NONE

Specifies processing level is none for long object.

**LOW** Specifies processing level is low for long object.

## LOB

### NORMAL

Specifies processing level is normal for LOB object. Default.

### NONE

Specifies processing level is none for LOB object.

**LOW** Specifies processing level is low for LOB object.

## XML

### NORMAL

Specifies processing level is normal for XML column object. Default. Pages of XML object will be checked for most inconsistencies. Actual XML data will not be inspected.

### NONE

Specifies processing level is none for XML column object. XML object will not be inspected at all.

**LOW** Specifies processing level is low for XML column object. Pages of XML object will be checked for some inconsistencies. Actual XML data will not be inspected.

### Usage Notes:

1. For check operations on table objects, the level of processing can be specified for the objects. The default is NORMAL level, specifying NONE for an object excludes it. Specifying LOW will do subset of checks that are done for NORMAL.
2. The check database can be specified to start from a specific table space or from a specific table by specifying the ID value to identify the table space or the table.
3. The check table space can be specified to start from a specific table by specifying the ID value to identify the table.
4. The processing of table spaces will affect only the objects that reside in the table space.
5. The online inspect processing will access database objects using isolation level uncommitted read. COMMIT processing will be done during INSPECT processing. It is advisable to end the unit of work by issuing a COMMIT or ROLLBACK before invoking INSPECT.
6. The online inspect check processing will write out unformatted inspection data results to the results file specified. The file will be written out to the diagnostic

data directory path. If there is no error found by the check processing, this result output file will be erased at the end of INSPECT operation. If there are errors found by the check processing, this result output file will not be erased at the end of INSPECT operation. After check processing completes, to see inspection details, the inspection result data will require to be formatted out with the utility db2inspf. The results file will have file extension of the database partition number.

7. In a partitioned database environment, each database partition will generate its own results output file with extension corresponding to its database partition number. The output location for the results output file will be the database manager diagnostic data directory path. If the name of a file that already exists is specified, the operation will not be processed, the file will have to be removed before that file name can be specified.
8. Normal online inspect processing will access database objects using isolation level uncommitted read. Inserting a compression dictionary into the table will attempt to acquire write locks. Please refer to the ROWCOMPESTIMATE option for details on dictionary insert locking. Commit processing will be done during the inspect processing. It is advisable to end the unit of work by issuing a COMMIT or ROLLBACK before starting the inspect operation.

**Related reference:**

- “db2Inspect API - Inspect database for architectural integrity” in *Administrative API Reference*

## LIST ACTIVE DATABASES

Displays a subset of the information listed by the GET SNAPSHOT FOR ALL DATABASES command. An active database is available for connection and use by any application. 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 database partition that is listed in \$HOME/sql1lib/db2nodes.cfg. It returns the same information from any of these database partitions.

**Authorization:**

One of the following:

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

**Command syntax:**



**Command parameters:**

**AT DBPARTITIONNUM db-partition-number**

Specifies the database partition for which the status of the monitor switches is to be displayed.

**global** Returns an aggregate result for all nodes in a partitioned database system.

**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/

```

**Compatibilities:**

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.



**Related reference:**

- “GET SNAPSHOT ” on page 487
- “ACTIVATE DATABASE ” on page 330
- “DEACTIVATE DATABASE ” on page 411

---

## 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 database partition on which it is issued.

### Authorization:

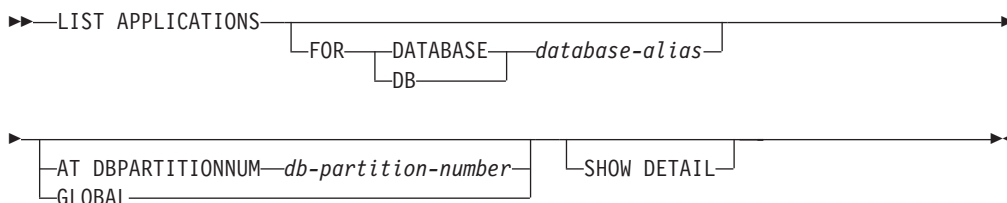
One of the following:

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

### Required connection:

Instance. To list applications for a remote instance, it is necessary to first attach to that instance.

### Command syntax:



### 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 database partition 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.

#### AT DBPARTITIONNUM *db-partition-number*

Specifies the database partition for which the status of the monitor switches is to be displayed.

**GLOBAL**

Returns an aggregate result for all database partitions in a partitioned database system.

**SHOW DETAIL**

Output will include the following additional information:

- Sequence #
- Application status
- Status change time
- Database path.

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 might wrap around when displayed on the screen.

**Examples:**

To list detailed information about the applications connected to the SAMPLE database, issue:

```
list applications for database sample show detail
```

**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 the GET SNAPSHOT command or the FORCE APPLICATION command 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.

**Compatibilities:**

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.

**Related reference:**

- "GET SNAPSHOT " on page 487
- "FORCE APPLICATION " on page 439
- "APPLICATIONS administrative view – Retrieve connected database application information" in *Administrative SQL Routines and Views*
- "FORCE APPLICATION command using the ADMIN\_CMD procedure" in *Administrative SQL Routines and Views*

## 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
-d	XML declarations	OFF
-e	Display SQLCODE/SQLSTATE	OFF
-f	Read from input file	OFF
-l	Log commands in history file	OFF
-n	Remove new line character	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

**Related reference:**

- “UPDATE COMMAND OPTIONS ” on page 768

## 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 database partitions.

If the ON *path* parameter is specified, the local database directory on that path is returned. This information is not the same at all database partitions.

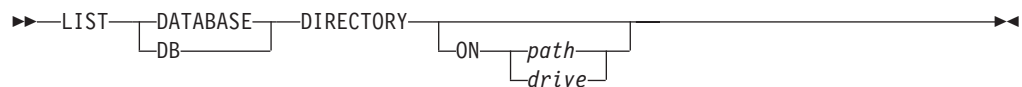
### Authorization:

None

### Required connection:

None. Directory operations affect the local directory only.

### Command syntax:



### 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. Note that the instance name is implied in the path. Please do not specify the instance name as part of the path.

### 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
Local database directory = /home/smith
Database release level = 8.00
Comment =
Directory entry type = Indirect
Catalog database partition number = 0
Alternate server hostname = montero
Alternate server port number = 29384

```

```
Database 2 entry:
```

```

Database alias = TC004000
Database name = TC004000
Node name = PRINODE
Database release level = a.00
Comment =

```

## LIST DATABASE DIRECTORY

```
Directory entry type = LDAP
Catalog database partition number = -1
Gateway node name = PRIGW
Alternate server node name =
Alternate server gateway node name = ALTGW
```

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 database partition number = 0
Database partition 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 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

The name of the directory 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.
- An LDAP entry indicates that the database location information is stored on an LDAP server.

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.

### **Principal name**

Specifies a fully qualified Kerberos principal name.

### **Catalog database partition number**

Specifies which node is the catalog database partition. This is the database partition on which the CREATE DATABASE command was issued.

### **Database partition number**

Specifies the number that is assigned in `db2nodes.cfg` to the node where the command was issued.

### **Alternate server hostname**

Specifies the host name or the IP address for the alternate server to be used when there is communication failure on the connection to the database. This field is displayed only for the system database directory.

### **Alternate server port number**

Specifies the port number for the alternate server to be used when there is communication failure on the connection to the database. This field is displayed only for the system database directory.

### **Alternate server node name**

If the directory entry type is LDAP, specifies the node name for the alternate server to be used when there is communication failure on the connection to the database.

### **Alternate server gateway node name**

If the directory entry type is LDAP, specifies the gateway node name for the alternate gateway to be used when there is communication failure on the connection to the database.

### **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.

### **Related reference:**

- "CHANGE DATABASE COMMENT " on page 390
- "CREATE DATABASE " on page 395
- "db2DbDirGetNextEntry API - Get the next system or local database directory entry" in *Administrative API Reference*
- "UPDATE ALTERNATE SERVER FOR DATABASE" on page 762

---

## LIST DATABASE PARTITION GROUPS

Lists all database partition groups associated with the current database.

**Scope:**

This command can be issued from any database partition that is listed in `$HOME/sql11ib/db2nodes.cfg`. It returns the same information from any of these database partitions.

**Authorization:**

For the system catalogs `SYSCAT.DBPARTITIONGROUPS` and `SYSCAT.DBPARTITIONGROUPDEF`, one of the following is required:

- *sysadm* or *dbadm* authority
- CONTROL privilege
- SELECT privilege.

**Required connection:**

Database

**Command syntax:**

```

>> LIST DATABASE PARTITION GROUPS [SHOW DETAIL]

```

**Command parameters:****SHOW DETAIL**

Specifies that the output should include the following information:

- Distribution map ID
- Database partition number
- In-use flag

**Examples:**

Following is sample output from the LIST DATABASE PARTITION GROUPS command:

```

DATABASE PARTITION GROUP NAME

IBMCACTGROUP
IBMDEFAULTGROUP

```

2 record(s) selected.

Following is sample output from the LIST DATABASE PARTITION GROUPS SHOW DETAIL command:

```

DATABASE PARTITION GROUP NAME PMAP_ID DATABASE PARTITION NUMBER IN_USE

IBMCACTGROUP 0 0 Y
IBMDEFAULTGROUP 1 0 Y

```

2 record(s) selected.

The fields are identified as follows:



## LIST DATABASE PARTITION GROUPS

### DATABASE PARTITION GROUP NAME

The name of the database partition group. The name is repeated for each database partition in the database partition group.

### PMAP\_ID

The ID of the distribution map. The ID is repeated for each database partition in the database partition group.

### DATABASE PARTITION NUMBER

The number of the database partition.

### IN\_USE

One of four values:

- Y** The database partition is being used by the database partition group.
- D** The database partition is going to be dropped from the database partition group as a result of a REDISTRIBUTE DATABASE PARTITION GROUP operation. When the operation completes, the database partition will not be included in reports from LIST DATABASE PARTITION GROUPS.
- A** The database partition has been added to the database partition group but is not yet added to the distribution map. The containers for the table spaces in the database partition group have been added on this database partition. The value is changed to Y when the REDISTRIBUTE DATABASE PARTITION GROUP operation completes successfully.
- T** The database partition has been added to the database partition group, but is not yet added to the distribution map. The containers for the table spaces in the database partition group have not been added on this database partition. Table space containers must be added on the new database partition for each table space in the database partition group. The value is changed to A when containers have successfully been added.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODEGROUPS can be substituted for DATABASE PARTITION GROUPS.

### Related reference:

- “REDISTRIBUTE DATABASE PARTITION GROUP ” on page 633

---

## LIST DBPARTITIONNUMS

Lists all database partitions associated with the current database.

**Scope:**

This command can be issued from any database partition that is listed in `$HOME/sql11ib/db2nodes.cfg`. It returns the same information from any of these database partitions.

**Authorization:**

None

**Required connection:**

Database

**Command syntax:**

▶▶—LIST DBPARTITIONNUMS—▶▶

**Command parameters:**

None

**Examples:**

Following is sample output from the LIST DBPARTITIONNUMS command:

```
DATABASE PARTITION NUMBER

 0
 2
 5
 7
 9
```

5 record(s) selected.

**Compatibilities:**

For compatibility with versions earlier than Version 8:

- The keyword `NODES` can be substituted for `DBPARTITIONNUMS`.

**Related reference:**

- “REDISTRIBUTE DATABASE PARTITION GROUP ” on page 633

## 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*
- *sysmon*

### 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

## LIST DCS APPLICATIONS

- Client protocol
- Client code page
- Process ID of the client application
- Host coded character set ID (CCSID).

### 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.

### 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.

### Related reference:

- "FORCE APPLICATION " on page 439

---

## 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 local 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.

**Comment**

Describes the database entry.

## LIST DCS DIRECTORY

### 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 the CATALOG DCS DATABASE command 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 databases on OS/390 and z/OS host
- DB2 databases on iSeries hosts
- DB2 databases on VSE & VM hosts

### Related reference:

- “sqllegdgt API - Get database connection services (DCS) directory entries” in *Administrative API Reference*
- “CATALOG DCS DATABASE ” on page 375

---

## LIST DRDA INDOUBT TRANSACTIONS

Provides a list of transactions that are indoubt between DRDA requesters and DRDA servers. If DRDA commit protocols are being used, lists indoubt transactions between DRDA sync point 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.

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).

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.

## LIST DRDA INDOUBT TRANSACTIONS

Before issuing the LIST DRDA INDOUBT TRANSACTIONS command, the application process must be connected to the DB2 sync point manager (SPM) instance. Use the *spm\_name* database manager configuration parameter as the *dbalias* on the CONNECT statement.

TCP/IP connections, using the SPM to coordinate commits, use DRDA two-phase commit protocols.

### Related tasks:

- “Resolving indoubt transactions manually” in *Administration Guide: Planning*

### Related reference:

- “sqlcspqy API - List DRDA indoubt transactions” in *Administrative API Reference*
- “sqlxhfrg API - Forget transaction status” in *Administrative API Reference*
- “sqlxphcm API - Commit an indoubt transaction” in *Administrative API Reference*
- “sqlxphrl API - Roll back an indoubt transaction” in *Administrative API Reference*



## 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, incremental backup, restore, and rollforward operations. Additional logged events include create, alter, drop, or rename table space, reorganize table, drop table, and load.

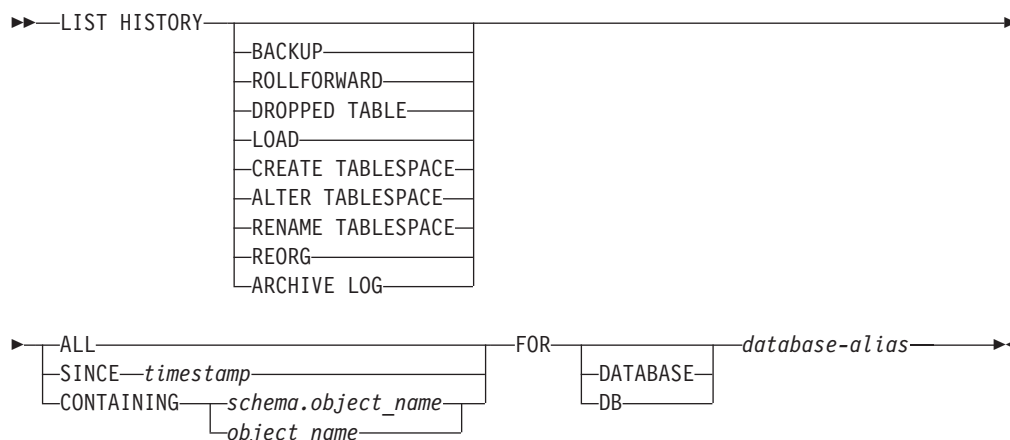
### Authorization:

None

### Required connection:

Instance. You must attach to any remote database in order to run this command against it. For a local database, an explicit attachment is not required.

### Command syntax:



### Command parameters:

#### HISTORY

Lists all events that are currently logged in the history file.

#### BACKUP

Lists backup and restore operations.

#### ROLLFORWARD

Lists rollforward operations.

#### DROPPED TABLE

Lists dropped table records. A dropped table record is created only when the table is dropped and the table space containing it has the DROPPED TABLE RECOVERY option enabled. Returns the CREATE TABLE syntax for partitioned tables and indicates which table spaces contained data for the table that was dropped.

#### LOAD

Lists load operations.

#### CREATE TABLESPACE

Lists table space create and drop operations.

## LIST HISTORY

### RENAME TABLESPACE

Lists table space renaming operations.

### REORG

Lists reorganization operations. Includes information for each reorganized data partition of a partitioned table.

### ALTER TABLESPACE

Lists alter table space operations.

### ARCHIVE LOG

Lists archive log operations and the archived logs.

**ALL** Lists all entries of the specified type in the history file.

### SINCE **timestamp**

A complete time stamp (format *yyyymmddhhmmss*), or an initial prefix (minimum *yyyy*) can be specified. All entries with time stamps equal to or greater 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 SYSIBMADM.DB\_HISTORY administrative view can be used to retrieve data from all database partitions.

The report generated by this command contains the following symbols:

#### Operation

```
A - Create table space
B - Backup
C - Load copy
D - Dropped table
F - Roll forward
G - Reorganize table
L - Load
N - Rename table space
O - Drop table space
Q - Quiesce
R - Restore
T - Alter table space
U - Unload
X - Archive log
```

#### Type

#### Archive Log types:

```
P - Primary log path
M - Secondary (mirror) log path
N - Archive log command
F - Failover archive path
```

- 1 - Primary log archive method
- 2 - Secondary log archive method

**Backup types:**

- F - Offline
- N - Online
- I - Incremental offline
- O - Incremental online
- D - Delta offline
- E - Delta online
- R - Rebuild

**Rollforward types:**

- E - End of logs
- P - Point in time

**Load types:**

- I - Insert
- R - Replace

**Alter table space types:**

- C - Add containers
- R - Rebalance

**Quiesce types:**

- S - Quiesce share
- U - Quiesce update
- X - Quiesce exclusive
- Z - Quiesce reset

**Related concepts:**

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

**Related reference:**

- “DB\_HISTORY administrative view – Retrieve history file information” in *Administrative SQL Routines and Views*

---

## 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.

Forgetting a transaction releases resources held by a heuristically completed transaction (that is, one that has been committed or rolled back heuristically). 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:

None

### 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).

## LIST INDOUBT TRANSACTIONS

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.

The LIST INDOUBT TRANSACTIONS command returns *type* information to show the role of the database in each indoubt transaction:

- TM** Indicates the indoubt transaction is using the database as a transaction manager database.
- RM** Indicates the indoubt transaction is using the database as a resource manager, meaning that it is one of the databases participating in the transaction, but is not the transaction manager database.

### 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. An indoubt transaction can also exist in an MPP environment.

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 acknowledgment (m), or missing federated commit acknowledgment (d) can be committed.

Only transactions whose status is indoubt (i), missing federated rollback acknowledgment (b), or ended (e) can be rolled back.

Only transactions whose status is committed (c), rolled back (r), missing federated commit acknowledgment (d), or missing federated rollback acknowledgment (b) can be forgotten.

In the commit phase of a two-phase commit, the coordinator node waits for commit acknowledgments. If one or more nodes do not reply (for example, because of node failure), the transaction is placed in missing commit acknowledgment state.

Indoubt transaction information is valid only at the time that the command is issued. Once in interactive dialog mode, transaction status might 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.

### Related concepts:

## LIST INDOUBT TRANSACTIONS

- “Configuration considerations for XA transaction managers” in *Administration Guide: Planning*

### **Related tasks:**

- “Resolving indoubt transactions manually” in *Administration Guide: Planning*

### **Related reference:**

- “db2XaListIndTrans API - List indoubt transactions” in *Administrative API Reference*
- “sqlxhfrg API - Forget transaction status” in *Administrative API Reference*
- “sqlxphcm API - Commit an indoubt transaction” in *Administrative API Reference*
- “sqlxphrl API - Roll back an indoubt transaction” in *Administrative API Reference*

---

## LIST NODE DIRECTORY

Lists the contents of the node directory.

### Authorization:

None

### Required connection:

None

### Command syntax:

```

▶▶—LIST—[ADMIN]—NODE DIRECTORY—[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 = 2
```

```
Node 1 entry:
```

```

Node name = LANNODE
Comment =
Directory entry type = LDAP
Protocol = TCPIP
Hostname = LAN.db2ntd3.torolab.ibm.com
Service name = 50000

```

```
Node 2 entry:
```

```

Node name = TLBA10ME
Comment =
Directory entry type = LOCAL
Protocol = TCPIP
Hostname = tlba10me
Service name = 447

```

The following is sample output from LIST ADMIN NODE DIRECTORY:

```
Node Directory
```

```
Number of entries in the directory = 2
```

## LIST NODE DIRECTORY

Node 1 entry:

Node name	= LOCALADM
Comment	=
Directory entry type	= LOCAL
Protocol	= TCPIP
Hostname	= jaguar
Service name	= 523

Node 2 entry:

Node name	= MYDB2DAS
Comment	=
Directory entry type	= LDAP
Protocol	= TCPIP
Hostname	= peng.torolab.ibm.com
Service name	= 523

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.

### **Directory entry type**

LOCAL means the entry is found in the local node directory file. LDAP means the entry is found on the LDAP server or LDAP cache.

### **Protocol**

The communications protocol cataloged for the node.

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:

- LDAP
- Local
- Named pipe
- TCPIP
- TCPIP4
- TCPIP6



**Related reference:**

- “sqlengne API - Get the next node directory entry” in *Administrative API Reference*
- “CATALOG LDAP NODE ” on page 381
- “CATALOG LOCAL NODE ” on page 382
- “CATALOG NAMED PIPE NODE ” on page 384
- “CATALOG TCPIP/TCPIP4/TCPIP6 NODE ” on page 387

---

## 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. That name is used to access the database or file system through ODBC APIs. On Windows, 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 only.

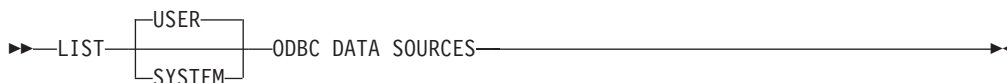
### Authorization:

None

### Required connection:

None

### Command syntax:



### 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
```

### Related reference:

- “CATALOG ODBC DATA SOURCE ” on page 386
- “UNCATALOG ODBC DATA SOURCE ” on page 753

## LIST PACKAGES/TABLES

Lists packages or tables associated with the current database.

### Authorization:

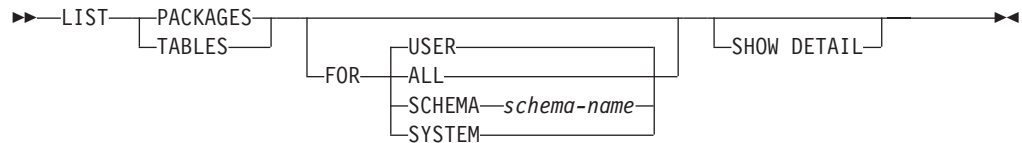
For the system catalog SYSCAT.PACKAGES (LIST PACKAGES) and SYSCAT.TABLES (LIST TABLES), one of the following is required:

- *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:



### 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

If this option is chosen with the LIST TABLES command, the full table name and schema name are displayed. If this option is not specified, the table name is truncated to 30 characters, and the ">" symbol in the 31st column represents the truncated portion of the table name; the schema name is truncated to 14 characters and the ">" symbol in the 15th column represents the truncated portion of the schema name. If this option is chosen with the LIST PACKAGES command, the full package schema (creator), version and boundby authid are displayed, and the package unique\_id (consistency token shown in hexadecimal form). If this option is not specified, the schema name and bound by ID are truncated to 8 characters and the ">" symbol in the 9th column represents the truncated portion of the schema or bound by ID; the version is truncated to 10 characters and the ">" symbol in the 11th column represents the truncated portion of the version.

## LIST PACKAGES/TABLES

### Examples:

The following is sample output from LIST PACKAGES:

Package	Schema	Version	Bound by	Total sections	Valid	Format	Isolation level	Blocking
F4INS	USERA	VER1	SNOWBELL	221	Y	0	CS	U
F4INS	USERA	VER2.0	SNOWBELL	201	Y	0	RS	U
F4INS	USERA	VER2.3	SNOWBELL	201	N	3	CS	U
F4INS	USERA	VER2.5	SNOWBELL	201	Y	0	CS	U
PKG12	USERA		USERA	12	Y	3	RR	B
PKG15	USERA		USERA	42	Y	3	RR	B
SALARY	USERT	YEAR2000	USERT	15	Y	3	CS	N

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 interface to the system tables.

The following SELECT statements return information found in the system tables. They can be expanded to select the additional information that the system tables provide.

```
select tabname, tabschema, type, create_time
from syscat.tables
order by tabschema, tabname;
```

```
select pkgname, pkgschema, pkgversion, unique_id, boundby, total_sect,
 valid, format, isolation, blocking
from syscat.packages
order by pkgschema, pkgname, pkgversion;
```

```
select tabname, tabschema, type, create_time
from syscat.tables
where tabschema = 'SYSCAT'
order by tabschema, tabname;
```

```
select pkgname, pkgschema, pkgversion, unique_id, boundby, total_sect,
 valid, format, isolation, blocking
from syscat.packages
where pkgschema = 'NULLID'
order by pkgschema, pkgname, pkgversion;
```

```
select tabname, tabschema, type, create_time
from syscat.tables
where tabschema = 'USER'
order by tabschema, tabname;
```

```
select pkgname, pkgschema, pkgversion, unique_id, boundby, total_sect,
 valid, format, isolation, blocking
from syscat.packages
where pkgschema = USER
order by pkgschema, pkgname, pkgversion;
```

**Related concepts:**

- “Catalog views” in *SQL Reference, Volume 1*
- “Efficient SELECT statements” in *Performance Guide*

---

# LIST TABLESPACE CONTAINERS

Lists containers for the specified table space.

The table space snapshot contains all of the information displayed by the LIST TABLESPACE CONTAINERS command.

### 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 the LIST TABLESPACES command.

#### 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
```

## LIST TABLESPACE CONTAINERS

```
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
```

### Related concepts:

- “Snapshot monitor” in *System Monitor Guide and Reference*

### Related reference:

- “sqlbtcq API - Get the query data for all table space containers” in *Administrative API Reference*
- “LIST TABLESPACES ” on page 550
- “CONTAINER\_UTILIZATION administrative view – Retrieve table space container and utilization information” in *Administrative SQL Routines and Views*
- “TBSP\_UTILIZATION administrative view – Retrieve table space configuration and utilization information” in *Administrative SQL Routines and Views*

---

## LIST TABLESPACES

Lists table spaces for the current database.

Information displayed by this command is also available in the table space snapshot.

### Scope:

This command returns information only for the node on which it is executed.

### Authorization:

One of the following:

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

### 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 or index data, 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. The db2tbst (Get Tablespace State) command 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 usable 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 = Database 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

```

## LIST TABLESPACES

```

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 = Database 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

```

## LIST TABLESPACES

```
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 partitioned database environment, this command does not return all the table spaces in the database. To obtain a list of all the table spaces, query SYSCAT.TABLESPACES.

## LIST TABLESPACES

During a table space rebalance, the number of usable pages includes pages for the newly added container, but these new pages are not reflected in the number of free pages until the rebalance is complete. When a table space rebalance is not in progress, the number of used pages plus the number of free pages equals the number of usable pages.

### Related reference:

- “sqlbmtsq API - Get the query data for all table spaces” in *Administrative API Reference*
- “LIST TABLESPACE CONTAINERS ” on page 548
- “db2tbst - Get table space state ” on page 285
- “CONTAINER\_UTILIZATION administrative view – Retrieve table space container and utilization information” in *Administrative SQL Routines and Views*
- “TBSP\_UTILIZATION administrative view – Retrieve table space configuration and utilization information” in *Administrative SQL Routines and Views*

---

## LIST UTILITIES

Displays to standard output the list of active utilities on the instance. The description of each utility can include attributes such as start time, description, throttling priority (if applicable), as well as progress monitoring information (if applicable).

**Scope:**

This command only returns information for the database partition on which it is issued.

**Authorization:**

One of the following:

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

**Required connection:**

Instance.

**Command syntax:**

```

▶▶ LIST UTILITIES [SHOW DETAIL] ▶▶

```

**Command parameters:****SHOW DETAIL**

Displays detailed progress information for utilities that support progress monitoring.

**Examples:**

A RUNSTATS invocation on table `some_table`:

```
LIST UTILITIES
```

```

ID = 1
Type = RUNSTATS
Database Name = PROD
Description = krrose.some_table
Start Time = 12/19/2003 11:54:45.773215
Priority = 10

```

Monitoring the performance of an offline database backup:

```
LIST UTILITIES SHOW DETAIL
```

```

ID = 2
Type = BACKUP
Database Name = SAMPLE
Description = offline db
Start Time = 10/30/2003 12:55:31.786115
Priority = 0
Progress Monitoring:
 Phase Number [CURRENT] = 1
 Description =

```

## LIST UTILITIES

```
Work Metric = BYTES
Total Work Units = 20232453
Completed Work Units = 230637
Start Time = 10/30/2003 12:55:31.786115
```

### Usage notes:

Use this command to monitor the status of running utilities. For example, you might use this utility to monitor the progress of an online backup. In another example, you might investigate a performance problem by using this command to determine which utilities are running. If the utility is suspected to be responsible for degrading performance then you might elect to throttle the utility (if the utility supports throttling). The ID from the LIST UTILITIES command is the same ID used in the SET UTIL\_IMPACT\_PRIORITY command.

### Related reference:

- “SET UTIL\_IMPACT\_PRIORITY” on page 724
- “SNAPUTIL administrative view and SNAP\_GET\_UTIL table function – Retrieve utility\_info logical data group snapshot information” in *Administrative SQL Routines and Views*
- “SNAPUTIL\_PROGRESS administrative view and SNAP\_GET\_UTIL\_PROGRESS table function – Retrieve progress logical data group snapshot information” in *Administrative SQL Routines and Views*

---

## LOAD

Loads data into a DB2 table. Data residing on the server can be in the form of a file, tape, or named pipe. Data residing on a remotely connected client can be in the form of a fully qualified file or named pipe. Data can also be loaded from a user-defined cursor or by using a user-written script or application. If the COMPRESS attribute for the table is set to YES, the data loaded will be subject to compression on every data and database partition for which a dictionary already exists in the table.

### Restrictions:

The load utility does not support loading data at the hierarchy level. The load utility is not compatible with range-clustered tables.

### Scope:

This command can be issued against multiple database partitions in a single request.

### 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 INSERT mode, TERMINATE mode (to terminate a previous load insert operation), or RESTART mode (to restart a previous load insert operation)
  - INSERT and DELETE privilege on the table when the load utility is invoked in REPLACE mode, TERMINATE mode (to terminate a previous load replace operation), or RESTART mode (to restart a previous load replace operation)
  - INSERT privilege on the exception table, if such a table is used as part of the load operation.
- To load data into a table that has protected columns, the session authorization ID must have LBAC credentials that allow write access to all protected columns in the table. Otherwise the load fails and an error (SQLSTATE 5U014) is returned.
- To load data into a table that has protected rows, the session authorization id must hold a security label that meets these criteria:
  - It is part of the security policy protecting the table
  - It was granted to the session authorization ID for write access or for all access

If the session authorization id does not hold such a security label then the load fails and an error (SQLSTATE 5U014) is returned. This security label is used to protect a loaded row if the session authorization ID's LBAC credentials do not allow it to write to the security label that protects that row in the data. This does not happen, however, when the security policy protecting the table was created with the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option of the CREATE SECURITY POLICY statement. In this case the load fails and an error (SQLSTATE 42519) is returned.

- If the REPLACE option is specified, the session authorization ID must have the authority to drop the table.

# LOAD

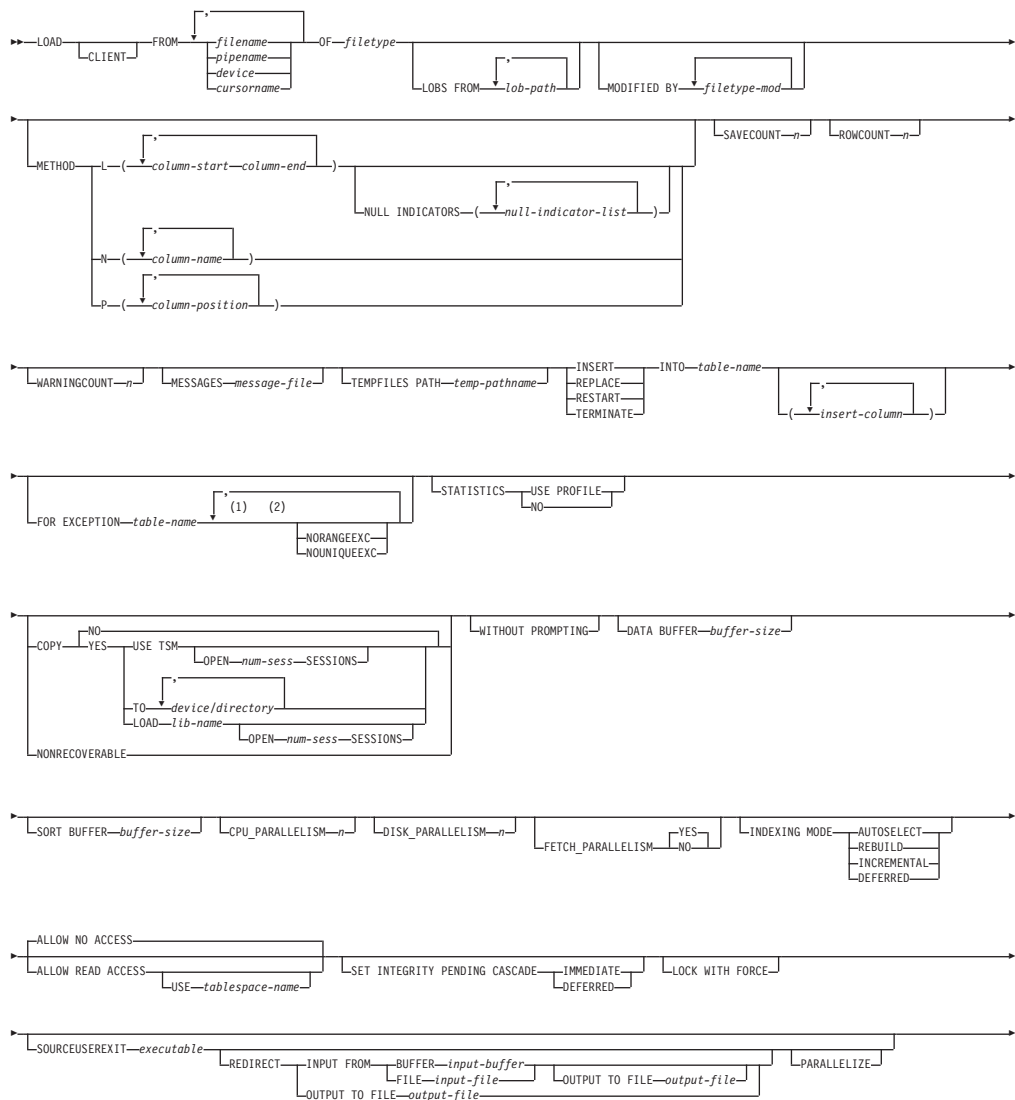
Since all load processes (and all DB2 server processes, in general) are owned by the instance owner, and all of these processes use the identification of the instance owner to access needed files, the instance owner must have read access to input data files. These input data files must be readable by 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:





**Notes:**

- 1 These keywords can appear in any order.
- 2 Each of these keywords can only appear once.

**Command parameters:****CLIENT**

Specifies that the data to be loaded resides on a remotely connected client. This option is ignored if the load operation is not being invoked from a remote client. This option is ignored if specified in conjunction with the CURSOR filetype.

**Notes:**

1. The `dumpfile` and `lobsinfile` modifiers refer to files on the server even when the CLIENT keyword is specified.
2. Code page conversion is not performed during a remote load operation. If the code page of the data is different from that of the server, the data code page should be specified using the `codepage` modifier.

In the following example, a data file (`/u/user/data.del`) residing on a remotely connected client is to be loaded into MYTABLE on the server database:

```
db2 load client from /u/user/data.del of del
modified by codepage=850 insert into mytable
```

**FROM filename/pipe/device/cursorname**

Specifies the file, pipe, device, or cursor referring to an SQL statement that contains the data being loaded. If the input source is a file, pipe, or device, it must reside on the database partition where the database resides, unless the CLIENT option is specified.

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. 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 database partition as the caller, relative paths can be used.
2. If data is exported into a file using the EXPORT command using the ADMIN\_CMD procedure, the data file is owned by the fenced user ID. This file is not usually accessible by the instance owner. To run the

## LOAD

LOAD from CLP or the ADMIN\_CMD procedure, the data file must be accessible by the instance owner ID, so read access to the data file must be granted to the instance owner.

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. (Multiple physical files would be considered logically one if they were all created with one invocation of the EXPORT command.)
4. If loading data that resides on a client machine, the data must be in the form of either a fully qualified file or a named pipe.

### OF filetype

Specifies the format of the data:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format)
- IXF (integrated exchange format, PC version), exported from the same or from another DB2 table
- CURSOR (a cursor declared against a SELECT or VALUES statement).

### LOBS FROM lob-path

The path to the data files containing LOB values to be loaded. The path must end with a slash (/). If the CLIENT option is specified, the path must be fully qualified. 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. The maximum number of paths that can be specified is 999. This will implicitly activate the LOBSINFILE behaviour.

This option is ignored when specified in conjunction with the CURSOR filetype.

### MODIFIED BY filetype-mod

Specifies file type modifier options. See File type modifiers for the load utility.

### METHOD

- L Specifies the start and end column numbers from which to load data. A column number is a byte offset from the beginning of a row of data. It is numbered starting from 1. This method can only be used with ASC files, and is the only valid method for that file type.

#### NULL INDICATORS null-indicator-list

This option can only be used when the METHOD L parameter is specified; that is, the input file is an ASC file). The null indicator list is a comma-separated list of positive integers specifying the column number of each null indicator field. The column number is the byte offset of the null indicator field from the beginning of a row of data. There must be one entry in the null indicator list for each data field defined in the METHOD L parameter. A column number of zero indicates that the corresponding data field always contains data.

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.

- 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 table column that is not nullable should have a corresponding entry in the METHOD N list. For example, given data fields 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 (F2, F1, F4, F3) is a valid request, while method N (F2, F1) is not valid. This method can only be used with file types IXF or CURSOR.
- P** Specifies the field numbers (numbered from 1) of the input data fields to be loaded. Each table column that is not nullable should have a corresponding entry in the METHOD P list. For example, given data fields 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 (2, 1, 4, 3) is a valid request, while method P (2, 1) is not valid. This method can only be used with file types IXF, DEL, or CURSOR, and is the only valid method for the DEL file type.

#### **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. 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.

This option is ignored when specified in conjunction with the CURSOR filetype.

#### **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.

#### **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 the load file or the target table is specified incorrectly, the load utility will generate a warning for each row that it attempts to load, which will cause the load to fail. 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.

#### **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

## LOAD

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.

The message file is usually populated with messages at the end of the load operation and, as such, is not suitable for monitoring the progress of the operation.

### **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 database partition.

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:

- 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.

### **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.

### **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.

### **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.

### **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 might 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.

The load terminate option will not remove a backup pending state from table spaces.

### **INTO table-name**

Specifies the database table into which the data is to be loaded. This table cannot be a system table or a declared temporary 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 SCHEMA.

#### **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 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 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)
```

#### **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. If an unqualified table name is specified, the table will be qualified with the CURRENT SCHEMA.

Information that is written to the exception table is *not* written to the dump file. In a partitioned database environment, an exception table must be defined for those database partitions 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.

#### **NORANGEEXC**

Indicates that if a row is rejected because of a range violation it will not be inserted into the exception table.

#### **NOUNIQUEEXC**

Indicates that if a row is rejected because it violates a unique constraint it will not be inserted into the exception table.

#### **STATISTICS USE PROFILE**

Instructs load to collect statistics during the load according to the profile defined for this table. This profile must be created before load is executed. The profile is created by the RUNSTATS command. If the profile does not exist and load is instructed to collect statistics according to the profile, a warning is returned and no statistics are collected.

#### **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.

#### **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 COPY NO option will also put the table space state into the Load in Progress table space state. This is a transient state that will disappear when the load completes or aborts. The data in any table in the table space cannot be updated or deleted until a table space backup or a full database backup is made. However, it is possible to access the data in any table by using the SELECT statement.

LOAD with COPY NO on a recoverable database leaves the table spaces in a backup pending state. For example, performing a LOAD with COPY NO

## LOAD

and INDEXING MODE DEFERRED will leave indexes needing a refresh. Certain queries on the table might require an index scan and will not succeed until the indexes are refreshed. The index cannot be refreshed if it resides in a table space which is in the backup pending state. In that case, access to the table will not be allowed until a backup is taken. Index refresh is done automatically by the database when the index is accessed by a query.

### **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).

### **USE TSM**

Specifies that the copy will be stored using Tivoli Storage Manager (TSM).

### **OPEN num-sess SESSIONS**

The number of I/O sessions to be used with TSM 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.

### **LOAD lib-name**

The name of the shared library (DLL on Windows operating systems) 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 where the user exit programs reside.

### **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 roll forward 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 operation is completed, such a table can only be dropped or restored from a backup (full or table space) taken after a commit point following the completion of the non-recoverable load operation.

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.

### **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.

### **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.

#### **SORT BUFFER buffer-size**

This option specifies a value that overrides the SORTHEAP database configuration parameter during a load operation. It is relevant only when loading tables with indexes and only when the INDEXING MODE parameter is not specified as DEFERRED. The value that is specified cannot exceed the value of SORTHEAP. This parameter is useful for throttling the sort memory that is used when loading tables with many indexes without changing the value of SORTHEAP, which would also affect general query processing.

#### **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 (usually based on the number of CPUs available) 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.

#### **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.

#### **FETCH\_PARALLELISM YES/NO**

When performing a load from a cursor where the cursor is declared using the DATABASE keyword, or when using the API `sqlu_remotefetch_entry` media entry, and this option is set to YES, the load utility attempts to parallelize fetching from the remote data source if possible. If set to NO, no parallel fetching is performed. The default value is YES. For more information, see Moving data using the CURSOR file type.

#### **INDEXING MODE**

Specifies whether the load utility is to rebuild indexes or to extend them incrementally. Valid values are:

## LOAD

### AUTOSELECT

The load utility will automatically decide between REBUILD or INCREMENTAL mode. The decision is based on the amount of data being loaded and the depth of the index tree. Information relating to the depth of the index tree is stored in the index object. RUNSTATS is not required to populate this information. AUTOSELECT is the default indexing 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 are true:

- The LOAD COPY option is specified (*logarchmeth1* with the USEREXIT or LOGRETAIN option).
- 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 might force a rebuild, or indexes might 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.

### ALLOW NO ACCESS

Load will lock the target table for exclusive access during the load. The table state will be set to Load In Progress during the load. ALLOW NO ACCESS is the default behavior. It is the only valid option for LOAD REPLACE.



When there are constraints on the table, the table state will be set to Set Integrity Pending as well as Load In Progress. The SET INTEGRITY statement must be used to take the table out of Set Integrity Pending state.

### ALLOW READ ACCESS

Load will lock the target table in a share mode. The table state will be set to both Load In Progress and Read Access. Readers can access the non-delta portion of the data while the table is being load. In other words, data that existed before the start of the load will be accessible by readers to the table, data that is being loaded is not available until the load is complete. LOAD TERMINATE or LOAD RESTART of an ALLOW READ ACCESS load can use this option; LOAD TERMINATE or LOAD RESTART of an ALLOW NO ACCESS load cannot use this option. Furthermore, this option is not valid if the indexes on the target table are marked as requiring a rebuild.

When there are constraints on the table, the table state will be set to Set Integrity Pending as well as Load In Progress, and Read Access. At the end of the load, the table state Load In Progress will be removed but the table states Set Integrity Pending and Read Access will remain. The SET INTEGRITY statement must be used to take the table out of Set Integrity Pending. While the table is in Set Integrity Pending and Read Access states, the non-delta portion of the data is still accessible to readers, the new (delta) portion of the data will remain inaccessible until the SET INTEGRITY statement has completed. A user can perform multiple loads on the same table without issuing a SET INTEGRITY statement. Only the original (checked) data will remain visible, however, until the SET INTEGRITY statement is issued.

ALLOW READ ACCESS also supports the following modifiers:

#### USE *tablespace-name*

If the indexes are being rebuilt, a shadow copy of the index is built in table space *tablespace-name* and copied over to the original table space at the end of the load during an INDEX COPY PHASE. Only system temporary table spaces can be used with this option. If not specified then the shadow index will be created in the same table space as the index object. If the shadow copy is created in the same table space as the index object, the copy of the shadow index object over the old index object is instantaneous. If the shadow copy is in a different table space from the index object a physical copy is performed. This could involve considerable I/O and time. The copy happens while the table is offline at the end of a load during the INDEX COPY PHASE.

Without this option the shadow index is built in the same table space as the original. Since both the original index and shadow index by default reside in the same table space simultaneously, there might be insufficient space to hold both indexes within one table space. Using this option ensures that you retain enough table space for the indexes.

This option is ignored if the user does not specify INDEXING MODE REBUILD or INDEXING MODE AUTOSELECT. This option will also be ignored if INDEXING MODE AUTOSELECT is chosen and load chooses to incrementally update the index.

### SET INTEGRITY PENDING CASCADE

If LOAD puts the table into Set Integrity Pending state, the SET

## LOAD

INTEGRITY PENDING CASCADE option allows the user to specify whether or not Set Integrity Pending state of the loaded table is immediately cascaded to all descendants (including descendent foreign key tables, descendent immediate materialized query tables and descendent immediate staging tables).

### IMMEDIATE

Indicates that Set Integrity Pending state is immediately extended to all descendent foreign key tables, descendent immediate materialized query tables and descendent staging tables. For a LOAD INSERT operation, Set Integrity Pending state is not extended to descendent foreign key tables even if the IMMEDIATE option is specified.

When the loaded table is later checked for constraint violations (using the IMMEDIATE CHECKED option of the SET INTEGRITY statement), descendent foreign key tables that were placed in Set Integrity Pending Read Access state will be put into Set Integrity Pending No Access state.

### DEFERRED

Indicates that only the loaded table will be placed in the Set Integrity Pending state. The states of the descendent foreign key tables, descendent immediate materialized query tables and descendent immediate staging tables will remain unchanged.

Descendent foreign key tables might later be implicitly placed in Set Integrity Pending state when their parent tables are checked for constraint violations (using the IMMEDIATE CHECKED option of the SET INTEGRITY statement). Descendent immediate materialized query tables and descendent immediate staging tables will be implicitly placed in Set Integrity Pending state when one of its underlying tables is checked for integrity violations. A warning (SQLSTATE 01586) will be issued to indicate that dependent tables have been placed in Set Integrity Pending state. See the Notes section of the SET INTEGRITY statement in the SQL Reference for when these descendent tables will be put into Set Integrity Pending state.

If the SET INTEGRITY PENDING CASCADE option is not specified:

- Only the loaded table will be placed in Set Integrity Pending state. The state of descendent foreign key tables, descendent immediate materialized query tables and descendent immediate staging tables will remain unchanged, and can later be implicitly put into Set Integrity Pending state when the loaded table is checked for constraint violations.

If LOAD does not put the target table into Set Integrity Pending state, the SET INTEGRITY PENDING CASCADE option is ignored.

### LOCK WITH FORCE

The utility acquires various locks including table locks in the process of loading. Rather than wait, and possibly timeout, when acquiring a lock, this option allows load to force off other applications that hold conflicting locks on the target table. Applications holding conflicting locks on the system catalog tables will not be forced off by the load utility. Forced applications will roll back and release the locks the load utility needs. The load utility can then proceed. This option requires the same authority as the FORCE APPLICATIONS command (SYSADM or SYSCTRL).

ALLOW NO ACCESS loads might force applications holding conflicting locks at the start of the load operation. At the start of the load the utility can force applications that are attempting to either query or modify the table.

ALLOW READ ACCESS loads can force applications holding conflicting locks at the start or end of the load operation. At the start of the load the load utility can force applications that are attempting to modify the table. At the end of the load operation, the load utility can force applications that are attempting to either query or modify the table.

#### **SOURCEUSEREXIT***executable*

Specifies an executable filename which will be called to feed data into the utility.

#### **REDIRECT**

##### **INPUT FROM**

###### **BUFFER** *input-buffer*

The stream of bytes specified in *input-buffer* is passed into the STDIN file descriptor of the process executing the given executable.

###### **FILE** *input-file*

The contents of this client-side file are passed into the STDIN file descriptor of the process executing the given executable.

##### **OUTPUT TO**

###### **FILE** *output-file*

The STDOUT and STDERR file descriptors are captured to the fully qualified server-side file specified.

#### **PARALLELIZE**

Increases the throughput of data coming into the load utility by invoking multiple user exit processes simultaneously. This option is only applicable in multi-partition database environments and is ignored in single-partition database environments.

For more information, see *Moving data using a customized application (user exit)*.

#### **PARTITIONED DB CONFIG**

Allows you to execute a load into a table distributed across multiple database partitions. The PARTITIONED DB CONFIG parameter allows you to specify partitioned database-specific configuration options. The partitioned-db-option values can be any of the following:

```

PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x

```

## LOAD

```
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x
```

Detailed descriptions of these options are provided in Load configuration options for partitioned database environments.

### RESTARTCOUNT

Reserved.

### USING directory

Reserved.

### 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 reflen=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 reflen=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.

## LOAD

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.

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 a Table with an Identity Column)

TABLE1 has 4 columns:

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 is the same as TABLE1, except that C2 is a GENERATED ALWAYS identity column.

Data records in DATAFILE1 (DEL format):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

Data records in DATAFILE2 (DEL format):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

#### Notes:

1. The following command generates identity values for rows 1 and 2, since no identity values are supplied in DATAFILE1 for those rows. Rows 3 and 4, however, are assigned the user-supplied identity values of 100 and 101, respectively.
 

```
db2 load from datafile1.del of del replace into table1
```
2. To load DATAFILE1 into TABLE1 so that identity values are generated for all rows, issue one of the following commands:
 

```
db2 load from datafile1.del of del method P(1, 3, 4)
 replace into table1 (c1, c3, c4)
db2load from datafile1.del of del modified by identityignore
 replace into table1
```
3. To load DATAFILE2 into TABLE1 so that identity values are generated for each row, issue one of the following commands:
 

```
db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
db2 load from datafile2.del of del modified by identitymissing
 replace into table1
```
4. To load DATAFILE1 into TABLE2 so that the identity values of 100 and 101 are assigned to rows 3 and 4, issue the following command:
 

```
db2 load from datafile1.del of del modified by identityoverride
 replace into table2
```

In this case, rows 1 and 2 will be rejected, because the utility has been instructed to override system-generated identity values in favor of user-supplied values. If user-supplied values are not present, however, the row must be rejected, because identity columns are implicitly not NULL.

5. If DATAFILE1 is loaded into TABLE2 without using any of the identity-related file type modifiers, rows 1 and 2 will be loaded, but rows 3 and 4 will be rejected, because they supply their own non-NULL values, and the identity column is GENERATED ALWAYS.

#### Example 5 (Loading using the CURSOR filetype)

Table ABC.TABLE1 has 3 columns:

```
ONE INT
TWO CHAR(10)
THREE DATE
```

Table ABC.TABLE2 has 3 columns:

```
ONE VARCHAR
TWO INT
THREE DATE
```

## LOAD

Executing the following commands will load all the data from ABC.TABLE1 into ABC.TABLE2:

```
db2 declare mycurs cursor for select two,one,three from abc.table1
db2 load from mycurs of cursor insert into abc.table2
```

If ABC.TABLE1 resides in a database different from the database ABC.TABLE2 is in, the DATABASE, USER, and USING options of the **DECLARE CURSOR** command can be used to perform the load. For example, if ABC.TABLE1 resides in database DB1, and the user ID and password for DB1 are user1 and pwd1 respectively, executing the following commands will load all the data from ABC.TABLE1 into ABC.TABLE2:

```
db2 declare mycurs cursor database DB1 user user1 using pwd1
for select two,one,three from abc.table1
db2 load from mycurs of cursor insert into abc.table2
```

### 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 materialized query tables that are dependent on the tables being loaded. Tables that include referential or check constraints are placed in Set Integrity Pending state. Summary tables that are defined with REFRESH IMMEDIATE, and that are dependent on tables being loaded, are also placed in Set Integrity Pending state. Issue the SET INTEGRITY statement to take the tables out of Set Integrity Pending state. Load operations cannot be carried out on replicated materialized query tables.
- If a clustering index exists on the table, the data should be sorted on the clustering index prior to loading. Data does not need to be sorted prior to loading into a multidimensional clustering (MDC) table, however.
- If you specify an exception table when loading into a protected table, any rows that are protected by invalid security labels will be sent to that table. This might allow users that have access to the exception table to access to data that they would not normally be authorized to access. For better security be careful who you grant exception table access to, delete each row as soon as it is repaired and copied to the table being loaded, and drop the exception table as soon as you are done with it.
- Security labels in their internal format might contain newline characters. If you load the file using the DEL file format, those newline characters can be mistaken for delimiters. If you have this problem use the older default priority for delimiters by specifying the delprioritychar file type modifier in the LOAD command.
- For performing a load using the CURSOR filetype where the DATABASE keyword was specified during the **DECLARE CURSOR** command, the user ID and password used to authenticate against the database currently connected to (for the load) will be used to authenticate against the source database (specified by the DATABASE option of the **DECLARE CURSOR** command). If no user ID or password was specified for the connection to the loading database, a user ID and password for the source database must be specified during the **DECLARE CURSOR** command.

### Related concepts:

- “Load overview” in *Data Movement Utilities Guide and Reference*



- “Privileges, authorities, and authorizations required to use Load” in *Data Movement Utilities Guide and Reference*

**Related tasks:**

- “Loading data” in *Data Movement Utilities Guide and Reference*

**Related reference:**

- “QUIESCE TABLESPACES FOR TABLE ” on page 615
- “LOAD command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “Load - CLP examples” in *Data Movement Utilities Guide and Reference*
- “Load configuration options for partitioned database environments” in *Data Movement Utilities Guide and Reference*



SQL3501W The table space(s) in which the table resides will not be placed in backup pending state since forward recovery is disabled for the database.

SQL3109N The utility is beginning to load data from file "/u/mydir/data/staffbig.del"

SQL3500W The utility is beginning the "LOAD" phase at time "03-21-2002 11:31:16.597045".

SQL3519W Begin Load Consistency Point. Input record count = "0".

SQL3520W Load Consistency Point was successful.

SQL3519W Begin Load Consistency Point. Input record count = "104416".

SQL3520W Load Consistency Point was successful.

SQL3519W Begin Load Consistency Point. Input record count = "205757".

SQL3520W Load Consistency Point was successful.

SQL3519W Begin Load Consistency Point. Input record count = "307098".

SQL3520W Load Consistency Point was successful.

SQL3519W Begin Load Consistency Point. Input record count = "408439".

SQL3520W Load Consistency Point was successful.

SQL3532I The Load utility is currently in the "LOAD" phase.

Number of rows read	= 453376
Number of rows skipped	= 0
Number of rows loaded	= 453376
Number of rows rejected	= 0
Number of rows deleted	= 0
Number of rows committed	= 408439
Number of warnings	= 0

Tablestate:  
Load in Progress

### Usage Notes:

In addition to locks, the load utility uses table states to control access to the table. The **LOAD QUERY** command can be used to determine the table state; **LOAD QUERY** can be used on tables that are not currently being loaded. For a partitioned table, the state reported is the most restrictive of the corresponding visible data partition states. For example, if a single data partition is in the READ ACCESS state and all other data partitions are in NORMAL state, the load query operation returns the READ ACCESS state. A load operation will not leave a subset of data partitions in a state different from the rest of the table. The table states described by **LOAD QUERY** are as follows:

#### Normal

No table states affect the table.

#### Set Integrity Pending

The table has constraints which have not yet been verified. Use the SET INTEGRITY statement to take the table out of Set Integrity Pending state. The load utility places a table in Set Integrity Pending state when it begins a load operation on a table with constraints.

## LOAD QUERY

### Load in Progress

There is a load operation in progress on this table.

### Load Pending

A load operation has been active on this table but has been aborted before the data could be committed. Issue a `LOAD TERMINATE`, `LOAD RESTART`, or `LOAD REPLACE` command to bring the table out of this state.

### Read Access Only

The table data is available for read access queries. Load operations using the `ALLOW READ ACCESS` option place the table in read access only state.

### Reorg Pending

A reorg recommended `ALTER TABLE` statement has been executed on the table. A classic reorg must be performed before the table is accessible again.

### Unavailable

The table is unavailable. The table can only be dropped or restored from a backup. Rolling forward through a non-recoverable load operation will place a table in the unavailable state.

### Not Load Restartable

The table is in a partially loaded state that will not allow a load restart operation. The table will also be in load pending state. Issue a `LOAD TERMINATE` or a `LOAD REPLACE` command to bring the table out of the not load restartable state. A table is placed in not load restartable state when a rollforward operation is performed after a failed load operation that has not been successfully restarted or terminated, or when a restore operation is performed from an online backup that was taken while the table was in load in progress or load pending state. In either case, the information required for a load restart operation is unreliable, and the not load restartable state prevents a load restart operation from taking place.

### Unknown

The `LOAD QUERY` command is unable determine the table state.

The progress of a load operation can also be monitored with the `LIST UTILITIES` command.

### Related concepts:

- “Load overview” in *Data Movement Utilities Guide and Reference*
- “Monitoring a load operation in a partitioned database environment using the `LOAD QUERY` command” in *Data Movement Utilities Guide and Reference*
- “Table locking, table states and table space states” in *Data Movement Utilities Guide and Reference*

### Related reference:

- “`LIST UTILITIES`” on page 555

## MIGRATE DATABASE

Converts previous versions of DB2 databases to the formats corresponding to the release run by the instance.

The **db2ckmig** command must be issued prior to migrating the instance. The **db2imigr** command implicitly calls the **db2ckmig**. Backup all databases prior to migration, and prior to the installation of the current version of DB2 database product on Windows operating systems.

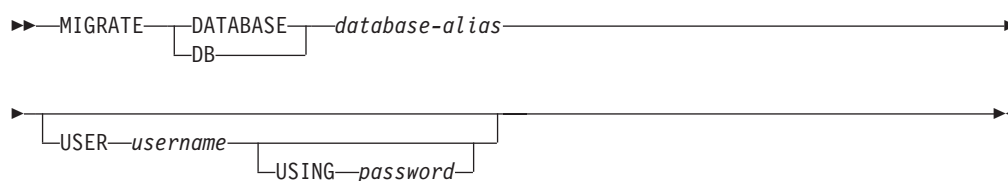
### Authorization:

*sysadm*

### Required connection:

This command establishes a database connection.

### Command syntax:



### 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.

If an error occurs during migration, it might be necessary to issue the **TERMINATE** command before attempting the suggested user response. For example, if a log full error occurs during migration (SQL1704: Database migration failed. Reason code "3"), it will be necessary to issue the **TERMINATE** command before increasing the

## MIGRATE DATABASE

values of the database configuration parameters LOGPRIMARY and LOGFILSIZ. The CLP must refresh its database directory cache if the migration failure occurs after the database has already been relocated (which is likely to be the case when a "log full" error returns).

### Related tasks:

- "Migrating databases" in *Migration Guide*

### Related reference:

- "TERMINATE " on page 744
- "sqlmgdb API - Migrate previous version of DB2 database to current version" in *Administrative API Reference*

## PING

Tests the network response time of the underlying connectivity between a client and a connected database server.

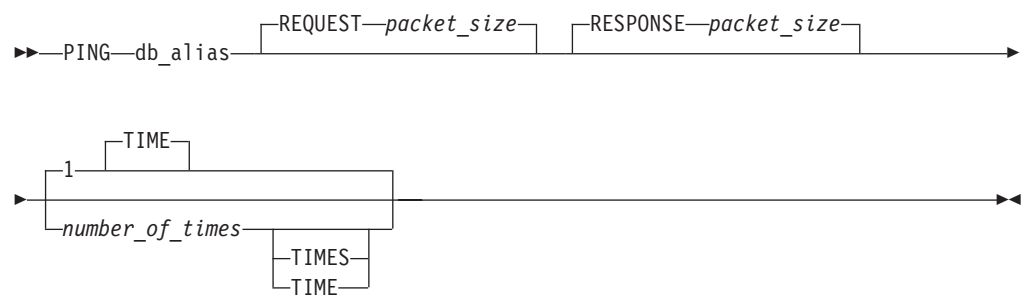
### Authorization:

None

### Required connection:

Database

### Command syntax:



### Command parameters:

#### db\_alias

Specifies the database alias for the database on a DRDA server that the ping is being sent to. This parameter, although mandatory, is not currently used. It is reserved for future use. Any valid database alias name can be specified.

#### REQUEST *packet\_size*

Specifies the size, in bytes, of the packet to be sent to the server. The size must be between 0 and 32767 inclusive. The default is 10 bytes. This option is only valid on servers running DB2 Database for Linux, UNIX, and Windows Version 8 or later, or DB2 UDB for z/OS Version 8 or later.

#### RESPONSE *packet\_size*

Specifies the size, in bytes, of the packet to be returned back to client. The size must be between 0 and 32767 inclusive. The default is 10 bytes. This option is only valid on servers running DB2 Database for Linux, UNIX, and Windows Version 8 or later, or DB2 UDB for z/OS Version 8 or later.

#### *number\_of\_times*

Specifies the number of iterations for this test. The value must be between 1 and 32767 inclusive. The default is 1. One timing will be returned for each iteration.

### Examples:

#### Example 1

To test the network response time for the connection to the host database *hostdb* once:

## PING

```
db2 ping hostdb 1
or
db2 ping hostdb
```

The command will display output that looks like this:

```
Elapsed time: 7221 microseconds
```

### Example 2

To test the network response time for the connection to the host database *hostdb* 5 times:

```
db2 ping hostdb 5
or
db2 ping hostdb 5 times
```

The command will display output that looks like this:

```
Elapsed time: 8412 microseconds
Elapsed time: 11876 microseconds
Elapsed time: 7789 microseconds
Elapsed time: 10124 microseconds
Elapsed time: 10988 microseconds
```

### Example 3

To test the network response time for a connection to the host database *hostdb*, with a 100-byte request packet and a 200-byte response packet:

```
db2 ping hostdb request 100 response 200
or
db2 ping hostdb request 100 response 200 1 time
```

### Usage notes:

A database connection must exist before invoking this command, otherwise an error will result.

The elapsed time returned is for the connection between the DB2 client and the DB2 server.

This command will not work when it is used from a DB2 Universal Database Version 7 client through a DB2 Connect Version 8 to a connected DB2 host database server.

### Related reference:

- “db2DatabasePing API - Ping the database to test network response time” in *Administrative API Reference*



## PRECOMPILE

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 database partition in `db2nodes.cfg`. In a partitioned database environment, it can be issued from any database partition server defined in the `db2nodes.cfg` file. It updates the database catalogs on the catalog database partition. Its effects are visible to all database partitions.

### 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.

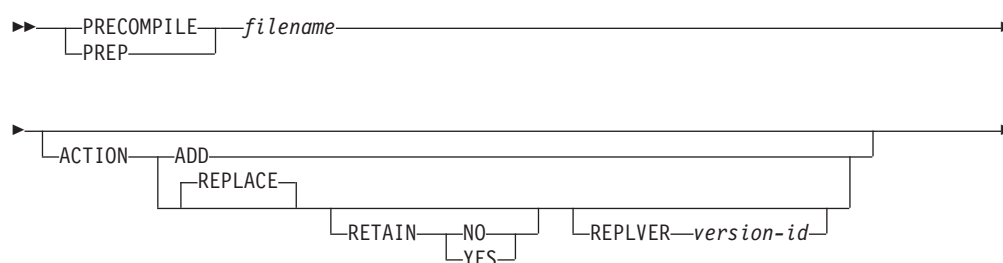
The user also needs all privileges required to compile any static SQL statements in the application. Privileges granted to groups are not used for authorization checking of static statements. If the user has *sysadm* authority, but not explicit privileges to complete the bind, the database manager grants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

### Required connection:

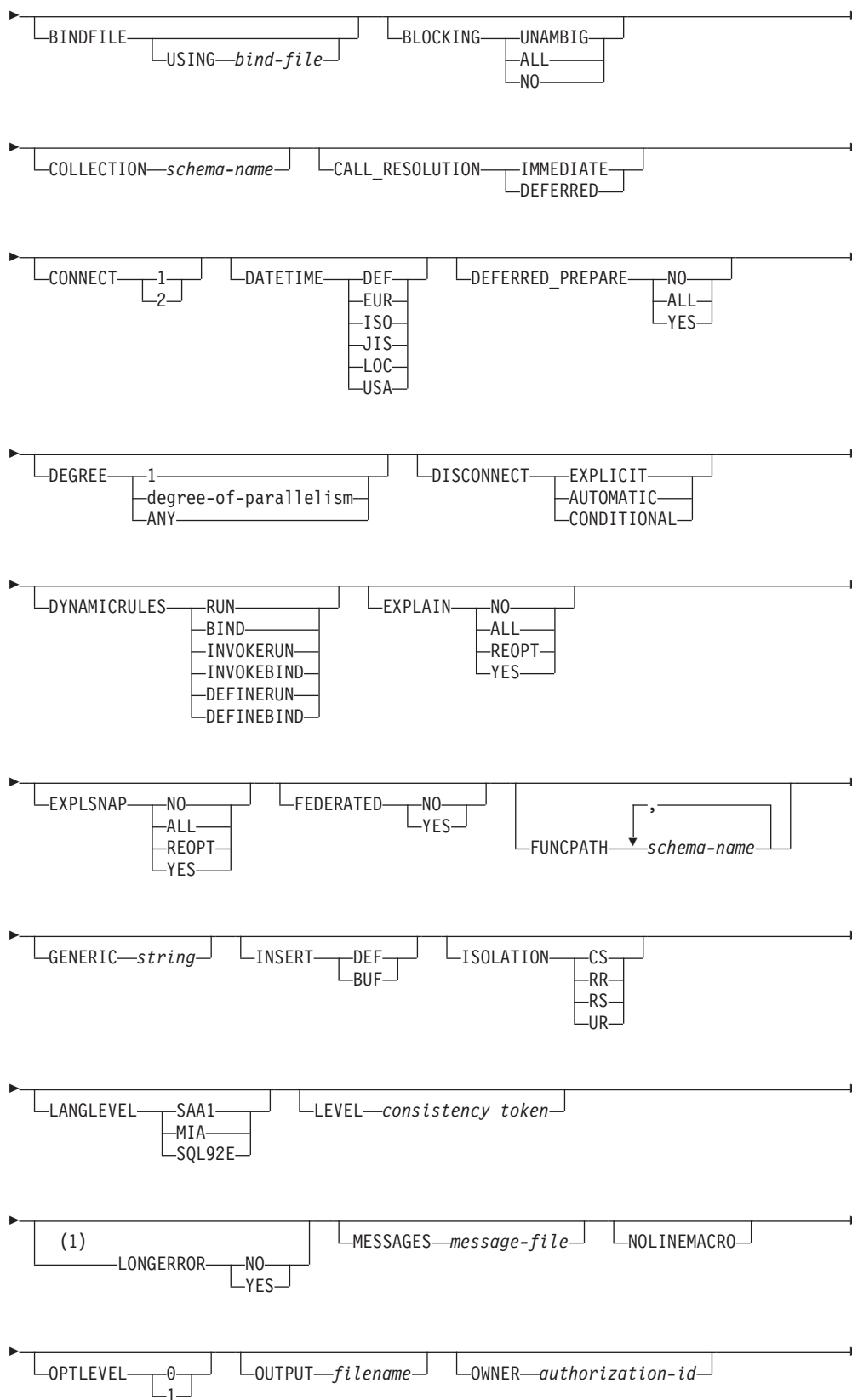
Database. If implicit connect is enabled, a connection to the default database is established.

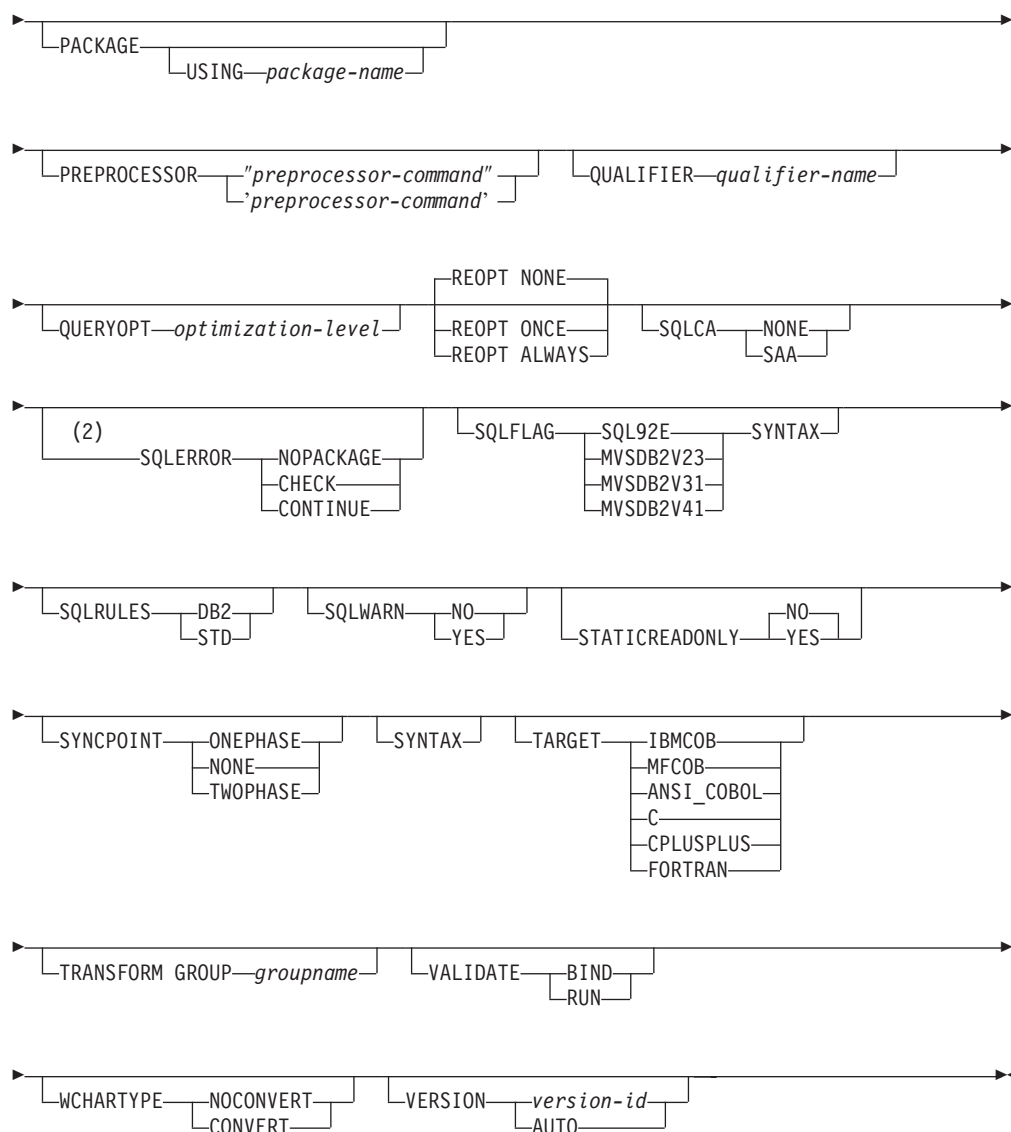
### Command syntax:

#### For DB2 for Windows and UNIX



# PRECOMPILE

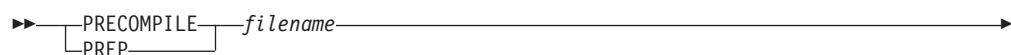




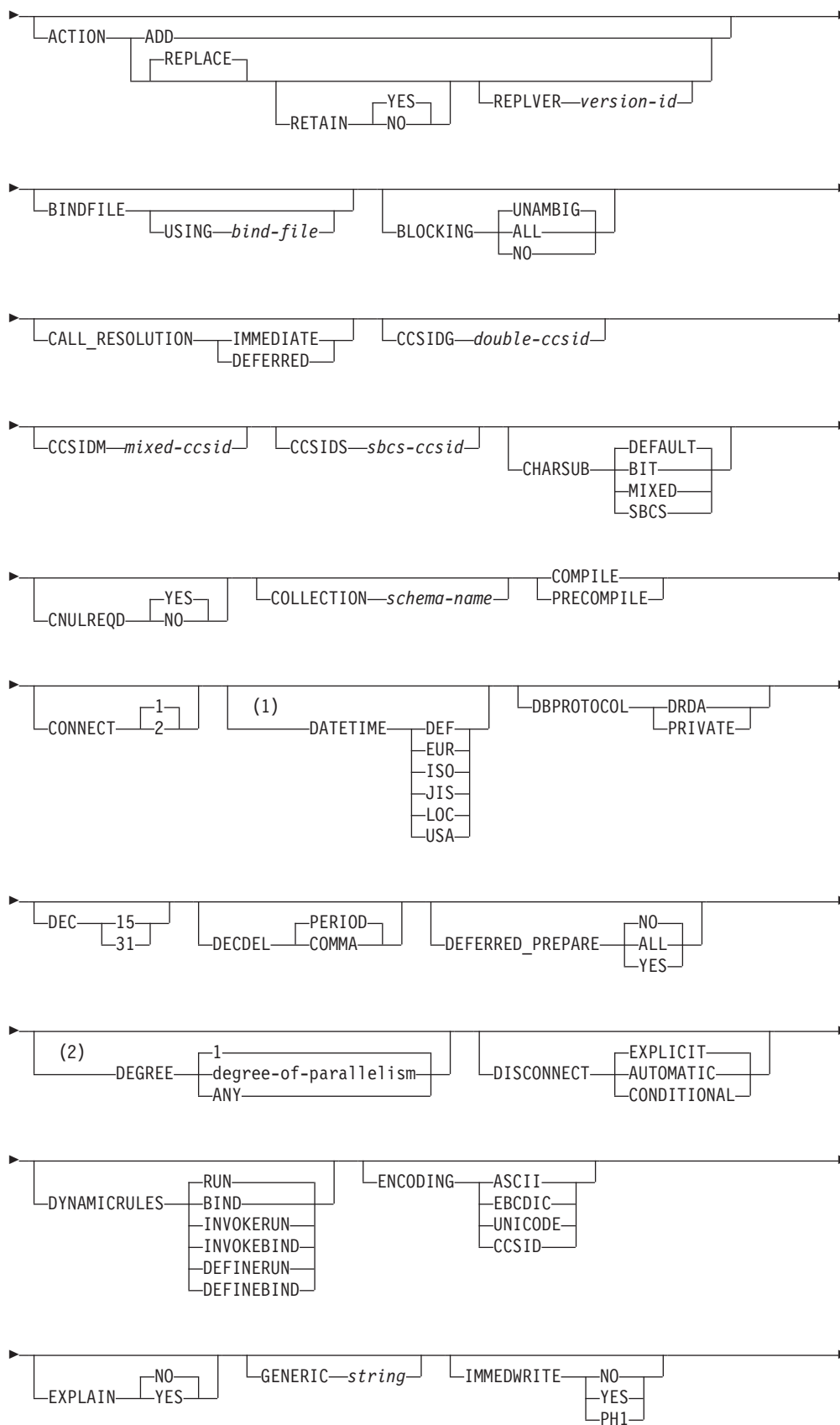
**Notes:**

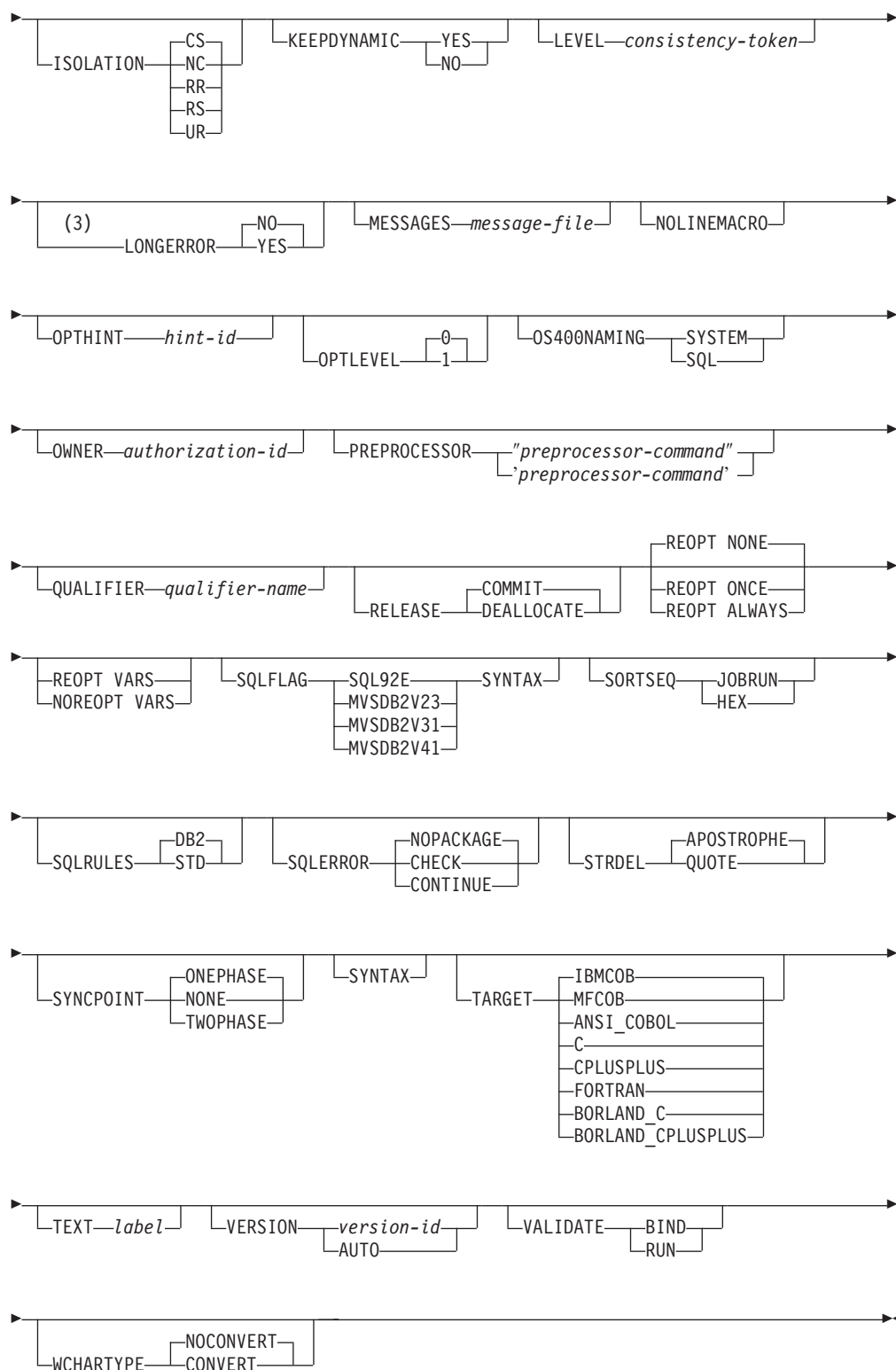
- 1 NO is the default for 32 bit systems and for 64 bit NT systems where long host variables can be used as declarations for INTEGER columns. YES is the default for 64 bit UNIX systems.
- 2 SYNTAX is a synonym for SQLERROR(CHECK).

**For DB2 on servers other than Windows and UNIX**



# PRECOMPILE





**Notes:**

- 1 If the server does not support the DATETIME DEF option, it is mapped to DATETIME ISO.
- 2 The DEGREE option is only supported by DRDA Level 2 Application Servers.

## PRECOMPILE

- 3 NO is the default for 32 bit systems and for 64 bit NT systems where long host variables can be used as declarations for INTEGER columns. YES is the default for 64 bit UNIX systems.

### 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 (Windows operating systems), or .sqC (UNIX based systems) must be specified for C++ applications (generates a .cxx file on Windows operating systems, or a .C file on UNIX based systems)
- .sqb must be specified for COBOL applications (generates a .cb1 file)
- .sqf must be specified for FORTRAN applications (generates a .for file on Windows operating systems, 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.

**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 existing package is to be replaced by a new one with the same package name and creator. This is the default value for the ACTION option.

#### 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. This value is not supported by DB2.

**YES** Preserves EXECUTE authorities when a package is replaced. This is the default value.

#### REPLVER version-id

Replaces a specific version of a package. The version identifier specifies which version of the package is to be replaced. If the specified version does not exist, an error is returned. If the REPLVER option of REPLACE is not specified, and a package already exists that matches the package name and version of the package being precompiled, that package will be replaced; if not, a new package will be added.

#### 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

Specifies the type of row blocking for cursors.

**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.

#### CALL\_RESOLUTION

If set, the CALL\_RESOLUTION DEFERRED option indicates that the CALL statement will be executed as an invocation of the deprecated sqlproc() API. If not set or if IMMEDIATE is set, the CALL statement will be executed as a normal SQL statement. SQL0204 will be issued if the precompiler fails to resolve the procedure on a CALL statement with CALL\_RESOLUTION IMMEDIATE.

#### 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 option is not supported by DB2 Database for Linux, UNIX, and Windows. 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 option is not supported by DB2 Database for Linux, UNIX, and Windows. The DRDA server will use a system defined default value if this option is not specified.

## PRECOMPILE

### CCSIDS *sbcsc-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 option is not supported by DB2 Database for Linux, UNIX, and Windows. 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 a 30-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.

**DEF** Use a date and time format associated with the territory 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 territory code of the database.
- USA** Use the IBM standard for U.S. date and time format.

**DBPROTOCOL**

Specifies what protocol to use when connecting to a remote site that is identified by a three-part name statement. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**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.

## PRECOMPILE

**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 for authorization checking of dynamic SQL statements. The authorization ID will also be used as the default package qualifier for implicit qualification of unqualified object references within dynamic SQL statements. 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.

### **DEFINERUN**

If the package is used within a routine context, the authorization ID of the routine definer is to be used for authorization checking and for implicit qualification of unqualified object references within dynamic SQL statements within the routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with **DYNAMICRULES RUN**.

### **DEFINEBIND**

If the package is used within a routine context, the authorization ID of the routine definer is to be used for authorization checking and for implicit qualification of unqualified object references within dynamic SQL statements within the routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with DYNAMICRULES BIND.

#### INVOKERUN

If the package is used within a routine context, the current statement authorization ID in effect when the routine is invoked is to be used for authorization checking of dynamic SQL statements and for implicit qualification of unqualified object references within dynamic SQL statements within that routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with DYNAMICRULES RUN.

#### INVOKEBIND

If the package is used within a routine context, the current statement authorization ID in effect when the routine is invoked is to be used for authorization checking of dynamic SQL statements and for implicit qualification of unqualified object references within dynamic SQL statements within that routine.

If the package is used as a standalone application, dynamic SQL statements are processed as if the package were bound with DYNAMICRULES BIND.

Because dynamic SQL statements will be using the authorization ID of the package owner in a package exhibiting bind behavior, the binder of the package should not have any authorities granted to them that the user of the package should not receive. Similarly, when defining a routine that will exhibit define behavior, the definer of the routine should not have any authorities granted to them that the user of the package should not receive since a dynamic statement will be using the authorization ID of the routine's definer.

The following dynamically prepared SQL statements cannot be used within a package that was not bound with DYNAMICRULES RUN: GRANT, REVOKE, ALTER, CREATE, DROP, COMMENT ON, RENAME, SET INTEGRITY, and SET EVENT MONITOR STATE.

#### ENCODING

Specifies the encoding for all host variables in static statements in the plan or package. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

#### 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 at prep/bind time for static statements and at run time for incremental bind statements.

If the package is to be used for a routine and the package contains incremental bind statements, then the routine must be defined as MODIFIES SQL DATA. If this is not done, incremental bind statements in the package will cause a run time error (SQLSTATE 42985).

### REOPT

Explain information for each reoptimizable incremental bind SQL statement will be placed in the Explain tables at run time. In addition, Explain information will be gathered for reoptimizable dynamic SQL statements at run time, even if the CURRENT EXPLAIN MODE special register is set to NO.

If the package is to be used for a routine, then the routine must be defined as MODIFIES SQL DATA, otherwise incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985).

**ALL** Explain information for each eligible static SQL statement will be placed in the Explain tables at prep/bind time. Explain information for each eligible incremental bind SQL statement will be placed in the Explain tables at run time. In addition, Explain information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN MODE special register is set to NO.

If the package is to be used for a routine, then the routine must be defined as MODIFIES SQL DATA, otherwise incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985).

### 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 at prep/bind time for static statements and at run time for incremental bind statements.

If the package is to be used for a routine and the package contains incremental bind statements, then the routine must be defined as MODIFIES SQL DATA or incremental bind statements in the package will cause a run time error (SQLSTATE 42985).

### REOPT

Explain Snapshot information for each reoptimizable incremental bind SQL statement will be placed in the Explain tables at run time. In addition, Explain Snapshot information will be gathered for reoptimizable dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT special register is set to NO.

If the package is to be used for a routine, then the routine must be defined as MODIFIES SQL DATA, otherwise incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985).

**ALL** An Explain Snapshot for each eligible static SQL statement will be placed in the Explain tables at prep/bind time. Explain Snapshot information for each eligible incremental bind SQL statement will be placed in the Explain tables at run time. In addition, Explain Snapshot information will be gathered for eligible dynamic SQL statements at run time, even if the CURRENT EXPLAIN SNAPSHOT special register is set to NO.

If the package is to be used for a routine, then the routine must be defined as MODIFIES SQL DATA, or incremental bind and dynamic statements in the package will cause a run time error (SQLSTATE 42985).

#### FEDERATED

Specifies whether a static SQL statement in a package references a nickname or a federated view. If this option is not specified and a static SQL statement in the package references a nickname or a federated view, a warning is returned and the package is created.

This option is not supported by DRDA servers.

**NO** A nickname or federated view is not referenced in the static SQL statements of the package. If a nickname or federated view is encountered in a static SQL statement during the prepare or bind phase of this package, an error is returned and the package is *not* created.

**YES** A nickname or federated view can be referenced in the static SQL statements of the package. If no nicknames or federated views are encountered in static SQL statements during the prepare or bind of the package, no errors or warnings are returned and the package is created.

#### 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

An 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.

#### INSERT

Allows a program being precompiled or bound against a DB2 Enterprise Server 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

Supports the binding of new options that are defined in the target database, but are not supported by DRDA. Do not use this option to pass bind options that *are* defined in BIND or PRECOMPILE. This option can substantially improve dynamic SQL performance. The syntax is as follows:

```
generic "option1 value1 option2 value2 ..."
```

## PRECOMPILE

Each option and value must be separated by one or more blank spaces. For example, if the target DRDA database is DB2 Universal Database, Version 8, one could use:

```
generic "explsnap all queryopt 3 federated yes"
```

to bind each of the EXPLSNAP, QUERYOPT, and FEDERATED options.

The maximum length of the string is 1023 bytes.

### IMMEDWRITE

Indicates whether immediate writes will be done for updates made to group buffer pool dependent pagesets or database partitions. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

### ISOLATION

Determines how far a program bound to this package can be isolated from the effect of other executing programs.

- 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 DRDA servers.

- 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 can 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).

**KEEPDYNAMIC**

Specifies whether dynamic SQL statements are to be kept after commit points. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

**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 option is not recommended for general use.

**LONGERROR**

Indicates whether long host variable declarations will be treated as an

## PRECOMPILE

error. For portability, `sqlint32` can be used as a declaration for an `INTEGER` column in precompiled C and C++ code.

- NO** Does not generate errors for the use of long host variable declarations. This is the default for 32 bit systems and for 64 bit NT systems where long host variables can be used as declarations for `INTEGER` columns. The use of this option on 64 bit UNIX platforms will allow long host variables to be used as declarations for `BIGINT` columns.
- YES** Generates errors for the use of long host variable declarations. This is the default for 64 bit UNIX systems.

### 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. This precompile option is used for the C/C++ programming languages only.

### OPTHINT

Controls whether query optimization hints are used for static SQL. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

### 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.

### OUTPUT filename

Overrides the default name of the modified source file produced by the compiler. It can include a path.

### OS400NAMING

Specifies which naming option is to be used when accessing DB2 UDB for iSeries data. Supported by DB2 UDB for iSeries only. For a list of supported option values, refer to the documentation for DB2 for iSeries.

Because of the slashes used as separators, a DB2 utility can still report a syntax error at execution time on certain SQL statements which use the iSeries system naming convention, even though the utility might have been



precompiled or bound with the OS400NAMING SYSTEM option. For example, the Command Line Processor will report a syntax error on an SQL CALL statement if the iSeries system naming convention is used, whether or not it has been precompiled or bound using the OS400NAMING SYSTEM option.

#### **OWNER authorization-id**

Designates a 30-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,

```
x1c -P -DMYMACRO=0
```

#### **QUALIFIER qualifier-name**

Provides an 30-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. The SET CURRENT QUERY OPTIMIZATION statement describes the complete range of optimization levels available. 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.

## PRECOMPILE

### REOPT

Specifies whether to have DB2 optimize an access path using values for host variables, parameter markers, and special registers. Valid values are:

#### NONE

The access path for a given SQL statement containing host variables, parameter markers or special registers will not be optimized using real values for these variables. The default estimates for these variables will be used instead, and this plan is cached and used subsequently. This is the default behavior.

**ONCE** The access path for a given SQL statement will be optimized using the real values of the host variables, parameter markers or special registers when the query is first executed. This plan is cached and used subsequently.

#### ALWAYS

The access path for a given SQL statement will always be compiled and reoptimized using the values of the host variables, parameter markers or special registers known at each execution time.

### REOPT / NOREOPT VARS

These options have been replaced by REOPT ALWAYS and REOPT NONE; however, they are still supported for compatibility with previous releases. Specifies whether to have DB2 determine an access path at run time using values for host variables, parameter markers, and special registers. Supported by DB2 for OS/390 only. For a list of supported option values, refer to the documentation for DB2 for OS/390.

### 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

Creates a package, even if errors occur when binding SQL statements. Those statements that failed to bind for authorization or existence reasons can be incrementally bound at execution time if **VALIDATE RUN** is also specified. Any attempt to execute them at run time generates an error (SQLCODE -525, SQLSTATE 51015).

#### 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.

**MVSD2V23 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.

**MVSD2V31 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.

**MVSD2V41 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.

**SORTSEQ**

Specifies which sort sequence table to use on the iSeries system. Supported by DB2 UDB for iSeries only. For a list of supported option values, refer to the documentation for DB2 for iSeries.

**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**

## PRECOMPILE

- 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).

**NO** Warnings will not be returned from the SQL compiler.

**YES** Warnings will be returned from the SQL compiler.

SQLCODE +238 is an exception. It is returned regardless of the **sqlwarn** option value.

### STATICREADONLY

Determines whether static cursors will be treated as being READ ONLY. This DB2 precompile/bind option is not supported by DRDA.

**NO** All static cursors will take on the attributes as would normally be generated given the statement text and the setting of the LANGLEVEL precompile option. This is the default value.

**YES** Any static cursor that does not contain the FOR UPDATE or FOR READ ONLY clause will be considered READ ONLY.

### 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. This command parameter is ignored and is only included here for backward compatibility.

#### 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.

**MFCOB**

Code is generated for the Micro Focus COBOL compiler. This is the default if a **target** value is not specified with the COBOL precompiler on all UNIX operating systems and Windows.

**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.

**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.

**TRANSFORM GROUP**

Specifies the transform group name to be used by static SQL statements for exchanging user-defined structured type values with host programs. This transform group is not used for dynamic SQL statements or for the exchange of parameters and results with external functions or methods. This option is not supported by DRDA servers.

**groupname**

An SQL identifier of up to 18 characters in length. A group name cannot include a qualifier prefix and cannot begin with the prefix SYS since this is reserved for database use. In a static SQL statement that interacts with host variables, the name of the transform group to be used for exchanging values of a structured type is as follows:

- The group name in the TRANSFORM GROUP bind option, if any
- The group name in the TRANSFORM GROUP prep option as specified at the original precompilation time, if any
- The DB2\_PROGRAM group, if a transform exists for the given type whose group name is DB2\_PROGRAM

## PRECOMPILE

- No transform group is used if none of the above conditions exist.

The following errors are possible during the bind of a static SQL statement:

- **SQLCODE** *yyy*, **SQLSTATE** *xxxxx*: A transform is needed, but no static transform group has been selected.
- **SQLCODE** *yyy*, **SQLSTATE** *xxxxx*: The selected transform group does not include a necessary transform (TO SQL for input variables, FROM SQL for output variables) for the data type that needs to be exchanged.
- **SQLCODE** *yyy*, **SQLSTATE** *xxxxx*: The result type of the FROM SQL transform is not compatible with the type of the output variable, or the parameter type of the TO SQL transform is not compatible with the type of the input variable.

In these error messages, *yyyyy* is replaced by the SQL error code, and *xxxxx* by the SQL state code.

### 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.

**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 can be redone at execution time.

### VERSION

Defines the version identifier for a package. If this option is not specified, the package version will be "" (the empty string).

#### **version-id**

Specifies a version identifier that is any alphanumeric value, \$, #, @, \_ , -, or ., up to 64 characters in length.

#### **AUTO**

The version identifier will be generated from the consistency token. If the consistency token is a timestamp (it will be if the **LEVEL** option is not specified), the timestamp is converted into ISO character format and is used as the version identifier.

### WCHARTYPE

Specifies the format for graphic data.

#### **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.

When a package exhibits bind behavior, the following will be true:

1. The implicit or explicit value of the `BIND` option `OWNER` will be used for authorization checking of dynamic SQL statements.
2. The implicit or explicit value of the `BIND` option `QUALIFIER` will be used as the implicit qualifier for qualification of unqualified objects within dynamic SQL statements.
3. The value of the special register `CURRENT SCHEMA` has no effect on qualification.

## PRECOMPILE

In the event that multiple packages are referenced during a single connection, all dynamic SQL statements prepared by those packages will exhibit the behavior as specified by the DYNAMICRULES option for that specific package and the environment they are used in.

If an SQL statement was found to be in error and the PRECOMPILE option SQLERROR CONTINUE was specified, the statement will be marked as invalid and another PRECOMPILE must be issued in order to change the state of the SQL statement. Implicit and explicit rebind will not change the state of an invalid statement in a package bound with VALIDATE RUN. A statement can change from static to incremental bind or incremental bind to static across implicit and explicit rebinds depending on whether or not object existence or authority problems exist during the rebind.

Binding a package with REOPT ONCE or REOPT ALWAYS might change static and dynamic statement compilation and performance.

### Related concepts:

- “Authorization considerations for dynamic SQL” in *Developing SQL and External Routines*
- “Effect of DYNAMICRULES bind option on dynamic SQL” in *Developing Embedded SQL Applications*
- “Performance improvements when using REOPT option of the BIND command” in *Developing Embedded SQL Applications*
- “WCHARTYPE precompiler option for graphic data in C and C++ embedded SQL applications” in *Developing Embedded SQL Applications*

### Related tasks:

- “Specifying row blocking to reduce overhead” in *Performance Guide*

### Related reference:

- “BIND” on page 355
- “Datetime values” in *SQL Reference, Volume 1*
- “SET CURRENT QUERY OPTIMIZATION statement” in *SQL Reference, Volume 2*
- “sqlprep API - Precompile application program” in *Administrative API Reference*



## PRUNE HISTORY/LOGFILE

Used to delete entries from the recovery history file or to delete log files from the active log file path. Deleting entries from the recovery history file might be necessary if the file becomes excessively large and the retention period is high.

### Authorization:

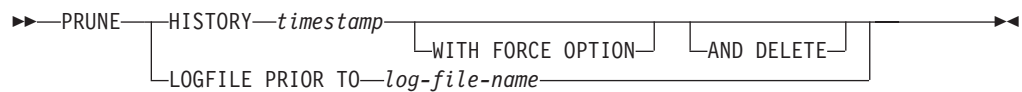
One of the following:

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

### Required connection:

Database

### Command syntax:



### 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 *yyyymmddhhmmss*), 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. A restore set is the most recent full database backup including any restores of that backup image. If this parameter is not specified, all entries from the backup image forward will be maintained in the history.

#### AND DELETE

Specifies that the associated log archives will be physically deleted (based on the location information) when the history file entry is removed. This option is especially useful for ensuring that archive storage space is recovered when log archives are no longer needed. If you are archiving logs via a user exit program, the logs cannot be deleted using this option.

#### LOGFILE PRIOR TO log-file-name

Specifies a string for a log file name, for example *S0000100.LOG*. All log files prior to (but not including) the specified log file will be deleted. The *LOGRETAIN* database configuration parameter must be set to *RECOVERY* or *CAPTURE*.

### Examples:

## PRUNE HISTORY/LOGFILE

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
```

199412 is interpreted as 19941201000000.

### Usage notes:

If the FORCE option is used, you might delete entries that are required for automatic restoration of databases. Manual restores will still work correctly. Use of this command can also prevent the **dbckrst** utility from being able to correctly analyze the complete chain of required backup images. Using the PRUNE HISTORY command without the FORCE option prevents required entries from being deleted.

Pruning backup entries from the history file causes related file backups on DB2 Data Links Manager servers to be deleted.

### Related concepts:

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

### Related reference:

- “PRUNE HISTORY/LOGFILE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

## PUT ROUTINE

Uses the specified routine SQL Archive (SAR) file to define a routine in the database.

### Authorization:

*dbadm*

### Required connection:

Database. If implicit connect is enabled, a connection to the default database is established.

### Command syntax:

```

▶▶ PUT ROUTINE FROM file-name
 ┌── OWNER new-owner ───┐
 └── USE REGISTERS ───┘

```

### Command parameters:

#### FROM *file-name*

Names the file where routine SQL archive (SAR) is stored.

#### OWNER *new-owner*

Specifies a new authorization name that will be used for authorization checking of the routine. The new owner must have the necessary privileges for the routine to be defined. If the OWNER clause is not specified, the authorization name that was originally defined for the routine is used.

#### USE REGISTERS

Indicates that the CURRENT SCHEMA and CURRENT PATH special registers are used to define the routine. If this clause is not specified, the settings for the default schema and SQL path are the settings used when the routine is defined. CURRENT SCHEMA is used as the schema name for unqualified object names in the routine definition (including the name of the routine) and CURRENT PATH is used to resolve unqualified routines and data types in the routine definition.

### Examples:

```
PUT ROUTINE FROM procs/proc1.sar;
```

### Usage Notes:

No more than one procedure can be concurrently installed under a given schema.

If a GET ROUTINE or a PUT ROUTINE operation (or their corresponding procedure) fails to execute successfully, it will always return an error (SQLSTATE 38000), along with diagnostic text providing information about the cause of the failure. For example, if the procedure name provided to GET ROUTINE does not identify an SQL procedure, diagnostic "-204, 42704" text will be returned, where "-204" and "42704" are the SQLCODE and SQLSTATE, respectively, that identify the cause of the problem. The SQLCODE and SQLSTATE in this example indicate that the procedure name provided in the GET ROUTINE command is undefined.

### Related reference:

## PUT ROUTINE

- “GET ROUTINE ” on page 485
- “PUT\_ROUTINE\_SAR procedure” in *Administrative SQL Routines and Views*

## 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_DBPARTITIONNUM = CATALOG_DBPARTITIONNUM
ATTACH_DBPARTITIONNUM = -1

```

If CONNECT\_DBPARTITIONNUM and ATTACH\_DBPARTITIONNUM 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\_DBPARTITIONNUM or the ATTACH\_DBPARTITIONNUM 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.

**Related reference:**

- “sqleqryc API - Query client connection settings” in *Administrative API Reference*
- “SET CLIENT ” on page 716

## QUIESCE

Forces all users off the specified instance and database and puts it into a quiesced mode. While the database instance or database is in quiesced mode, you can perform administrative tasks on it. After administrative tasks are complete, use the UNQUIESCE command to activate the instance and database and allow other users to connect to the database but avoid having to shut down and perform another database start.

In this mode, only users with authority in this restricted mode are allowed to attach or connect to the instance/database. Users with *sysadm*, *sysmaint*, and *sysctrl* authority always have access to an instance while it is quiesced, and users with *sysadm* and *dbadm* authority always have access to a database while it is quiesced.

**Scope:**

QUIESCE DATABASE results in all objects in the database being in the quiesced mode. Only the allowed user/group and *sysadm*, *sysmaint*, *dbadm*, or *sysctrl* will be able to access the database or its objects.

QUIESCE INSTANCE *instance-name* means the instance and the databases in the instance *instance-name* will be in quiesced mode. The instance will be accessible just for *sysadm*, *sysmaint*, and *sysctrl* and allowed user/group.

If an instance is in quiesced mode, a database in the instance cannot be put in quiesced mode.

**Authorization:**

One of the following:

For database level quiesce:

- *sysadm*
- *dbadm*

For instance level quiesce:

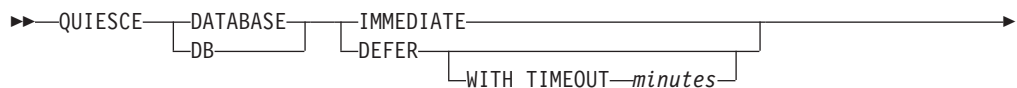
- *sysadm*
- *sysctrl*

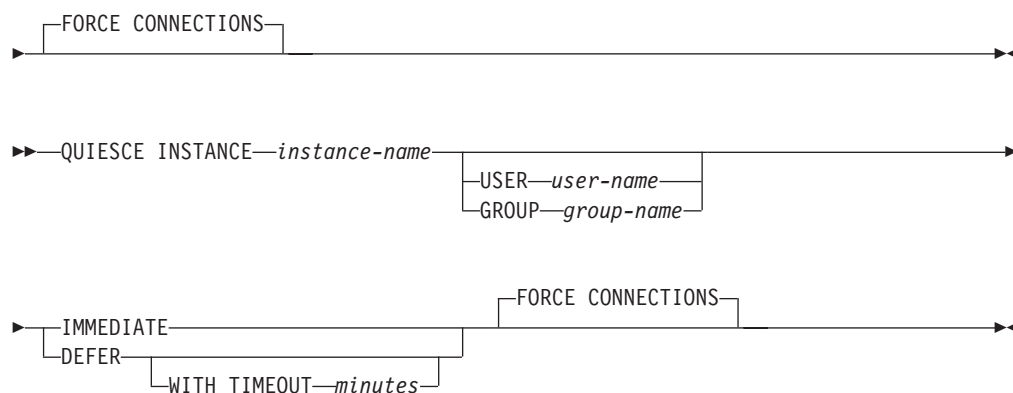
**Required connection:**

Database

(Database connection is not required for an instance quiesce.)

**Command syntax:**





**Command parameters:**

**DEFER**

Wait for applications until they commit the current unit of work.

**WITH TIMEOUT**

Specifies a time, in minutes, to wait for applications to commit the current unit of work. If no value is specified, in a single-partition database environment, the default value is 10 minutes. In a partitioned database environment the value specified by the *start\_stop\_timeout* database manager configuration parameter will be used.

**IMMEDIATE**

Do not wait for the transactions to be committed, immediately rollback the transactions.

**FORCE CONNECTIONS**

Force the connections off.

**DATABASE**

Quiesce the database. All objects in the database will be placed in quiesced mode. Only specified users in specified groups and users with *sysadm*, *sysmaint*, and *sysctrl* authority will be able to access to the database or its objects.

**INSTANCE *instance-name***

The instance *instance-name* and the databases in the instance will be placed in quiesced mode. The instance will be accessible only to users with *sysadm*, *sysmaint*, and *sysctrl* authority and specified users in specified groups.

**USER *user-name***

Specifies the name of a user who will be allowed access to the instance while it is quiesced.

**GROUP *group-name***

Specifies the name of a group that will be allowed access to the instance while the instance is quiesced.

**Examples:**

In the following example, the default behavior is to force connections, so it does not need to be explicitly stated and can be removed from this example.

```
db2 quiesce instance crankarm user frank immediate force connections
```

## QUIESCE

The following example forces off all users with connections to the database.

```
db2 quiesce db immediate
```

- The first example will quiesce the instance crankarm, while allowing user frank to continue using the database.

The second example will quiesce the database you are attached to, preventing access by all users except those with one of the following authorities: *sysadm*, *sysmaint*, *sysctrl*, or *dbadm*.

- This command will force all users off the database or instance if FORCE CONNECTION option is supplied. FORCE CONNECTION is the default behavior; the parameter is allowed in the command for compatibility reasons.
- The command will be synchronized with the FORCE and will only complete once the FORCE has completed.

### Usage notes:

- After QUIESCE INSTANCE, only users with *sysadm*, *sysmaint*, or *sysctrl* authority or a user name and group name provided as parameters to the command can connect to the instance.
- After QUIESCE DATABASE, users with *sysadm*, *sysmaint*, *sysctrl*, or *dbadm* authority, and GRANT/REVOKE privileges can designate who will be able to connect. This information will be stored permanently in the database catalog tables.

For example,

```
grant quiesce_connect on database to <username/groupname>
revoke quiesce_connect on database from <username/groupname>
```

### Related reference:

- “UNQUIESCE ” on page 754
- “db2DatabaseQuiesce API - Quiesce the database” in *Administrative API Reference*
- “db2InstanceQuiesce API - Quiesce instance” in *Administrative API Reference*
- “QUIESCE DATABASE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*



## 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
- 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 a partitioned database environment, this command acts locally on a database partition. It quiesces only that portion of table spaces belonging to the database partition on which the load operation is performed. For partitioned tables, all of the table spaces listed in SYSDATAPARTITIONS.TBSPACEID and SYSDATAPARTITIONS.LONG\_TBSPACEID associated with a table and with a status of normal, attached or detached, (for example, SYSDATAPARTITIONS.STATUS of 'N', 'A' or 'D', respectively) are quiesced.

### Authorization:

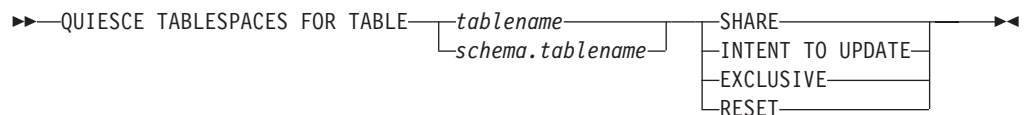
One of the following:

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

### Required connection:

Database

### Command syntax:



### 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 CURRENT SCHEMA will be used. The table cannot be a system catalog table.

## QUIESCE TABLESPACES FOR 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. A quiesce state cannot be reset if the connection that issued the quiesce request is still active.

### Example:

```
db2 quiesce tablespaces for table staff share
```

```
db2 quiesce tablespaces for table boss.org intent to update
```

### Usage notes:

This command is not supported for declared temporary tables.

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*. 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. The removal of a phantom quiesce requires a connection with the same user ID used when the quiesce mode was set.

To remove a phantom quiesce:

1. Connect to the database with the same user ID used when the quiesce mode was set.
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.

### Related reference:

- “sqluvqdp API - Quiesce table spaces for a table” in *Administrative API Reference*
- “LIST TABLESPACES ” on page 550
- “LOAD ” on page 557
- “QUIESCE ” on page 612
- “UNQUIESCE ” on page 754
- “QUIESCE TABLESPACES FOR TABLE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

## QUIT

---

## 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. The TERMINATE command does both.

**Related reference:**

- “TERMINATE ” on page 744

## REBIND

Allows the user to recreate a package stored in the database without the need for a bind file.

### Authorization:

One of the following:

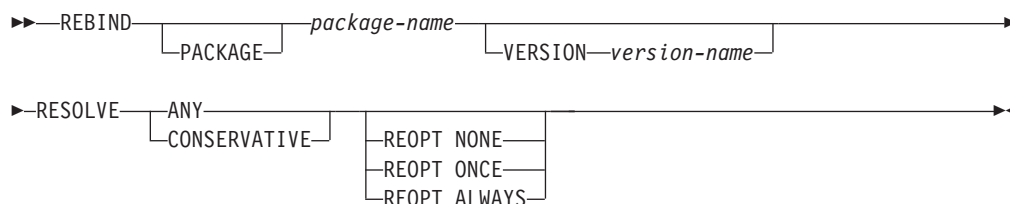
- *sysadm* or *dbadm* authority
- ALTERIN privilege on the schema
- BIND privilege on the package.

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. This default qualifier can 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:



### Command parameters:

#### PACKAGE package-name

The qualified or unqualified name that designates the package to be rebound.

#### VERSION version-name

The specific version of the package to be rebound. When the version is not specified, it is taken to be "" (the empty string).

#### RESOLVE

Specifies whether rebinding of the package is to be performed with or without conservative binding semantics. This affects whether new functions and data types are considered during function resolution and type resolution on static DML statements in the package. This option is not supported by DRDA. Valid values are:

**ANY** Any of the functions and types in the SQL path are considered for function and type resolution. Conservative binding semantics are not used. This is the default.

#### CONSERVATIVE

Only functions and types in the SQL path that were defined before

## REBIND

the last explicit bind time stamp are considered for function and type resolution. Conservative binding semantics are used. This option is not supported for an inoperative package.

### REOPT

Specifies whether to have DB2 optimize an access path using values for host variables, parameter markers, and special registers.

#### NONE

The access path for a given SQL statement containing host variables, parameter markers or special registers will not be optimized using real values for these variables. The default estimates for these variables will be used instead, and this plan is cached and used subsequently. This is the default behavior.

**ONCE** The access path for a given SQL statement will be optimized using the real values of the host variables, parameter markers or special registers when the query is first executed. This plan is cached and used subsequently.

#### ALWAYS

The access path for a given SQL statement will always be compiled and reoptimized using the values of the host variables, parameter markers or special registers known at each execution time.

### 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.

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 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 might 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 might 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 8 migration process. Given that this might 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).

If multiple versions of a package (many versions with the same package name and creator) exist, only one version can be rebound at once. If not specified in the VERSION option, the package version defaults to be "". Even if there exists only one package with a name that matches, it will not be rebound unless its version matches the one specified or the default.

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.

If an SQL statement was found to be in error and the BIND option SQLERROR CONTINUE was specified, the statement will be marked as invalid even if the problem has been corrected. REBIND will not change the state of an invalid statement. In a package bound with VALIDATE RUN, a statement can change from static to incremental bind or incremental bind to static across a REBIND depending on whether or not object existence or authority problems exist during the REBIND.

Rebinding a package with REOPT ONCE/ALWAYS might change static and dynamic statement compilation and performance.

## REBIND

If REOPT is not specified, REBIND will preserve the existing REOPT value used at precompile or bind time.

**Related reference:**

- “BIND” on page 355
- “RUNSTATS ” on page 702
- “db2rbind - Rebind all packages ” on page 230
- “sqlarbind API - Rebind package” in *Administrative API Reference*



## 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.

Reconcile produces a message file (reconcil.msg) in the instance path on UNIX based systems, and in the install path on Windows platforms. This file will contain warning and error messages that are generated during validation of the exception 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*). When no table is provided with the FOR EXCEPTION clause, a *.exp* file extension is appended to the exception report file.

#### FOR EXCEPTION *table-name*

Specifies the exception table into which rows that encounter link failures for DATALINK values are to be copied. If no table is specified, an exception report file is generated in the directory specified in the "DLREPORT" option.

### 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

## RECONCILE

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. A required DB2 Data Links Manager is one which has a referenced DATALINK value in the table. Reconcile tolerates the unavailability of a required DB2 Data Links Manager as well as other DB2 Data Links Managers that are configured to the database but are not part of the table data.

Reconciliation is performed with respect to all DATALINK data in the table. If file references cannot be reestablished, 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 a file is linked under a DATALINK column defined with WRITE PERMISSION ADMIN and modified but not yet committed (that is, the file is still in the update-in-progress state), the reconciliation process renames the modified file to a filename with .mod as the suffix. It also removes the file from the update-in-progress state. If the DATALINK column is defined with RECOVERY YES, the previous archive version is restored.

If an exception table is not specified, the host name, file name, column ID, and reason code for each of the DATALINK column values for which file references could not be reestablished are copied to an exception report file (<filename>.exp). If the file reference could not be reestablished because the DB2 Data Links Manager is unavailable or was dropped from the database using the DROP DATALINKS MANAGER command, the file name reported in the exception report file is not the full file name. The prefix will be missing. For example, if the original DATALINK value was `http://host.com/dlfs/x/y/a.b`, the value reported in the exception table will be `http://host.com/x/y/a.b`. The prefix name 'dlfs' will not be included.

If the DATALINK column is defined with RECOVERY YES, the previous archive version is restored.

At the end of the reconciliation process, the table is taken out of datalink reconcile pending (DRP) state only if reconcile processing is complete on all the required DB2 Data Links Managers. If reconcile processing is pending on any of the required DB2 Data Links Managers (because they were unavailable), the table will remain, or be placed, in DRP state. If for some reason, an exception occurred on one of the affected Data Links Managers such that the reconciliation could not be completed successfully, the table might also be placed in a DRNP state, for which further manual intervention will be required before full referential integrity for that table can be restored.

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 4 characters Right justified
 DATALINK columns

5 DATALINK column number 4 characters Right justified
 of the first violating padded with '0'
 column

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, READ PERMISSION FS, WRITE PERMISSION FS DATALINK  
           columns. Use update to relink the file.
- 00007-File is in unlink state.
- 00008-File restored but modified file has been copied to  
           <filename>.MOD
- 00009-File is already linked to another table.
- 00010-DB2 Data Links Manager referenced by the DATALINK  
           value has been dropped from the database using the  
           DROP DATALINKS MANAGER command.
- 00999-File could not be linked.

Example:

```
00001L000220002000400002000500001
```

- 00001 - Specifies that the number of violations is 1.
- L - Specifies that the type of violation is 'DATALINK violation'.

## RECONCILE

00022 - 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.

## RECOVER DATABASE

Restores and rolls forward a database to a particular point in time or to the end of the logs.

### Scope:

In a partitioned database environment, this command can only be invoked from the catalog partition. A database recover operation to a specified point in time affects all database partitions that are listed in the `db2nodes.cfg` file. A database recover operation to the end of logs affects the database partitions that are specified. If no partitions are specified, it affects all database partitions that are listed in the `db2nodes.cfg` file.

### Authorization:

To recover an existing database, one of the following:

- `sysadm`
- `sysctrl`
- `sysmaint`

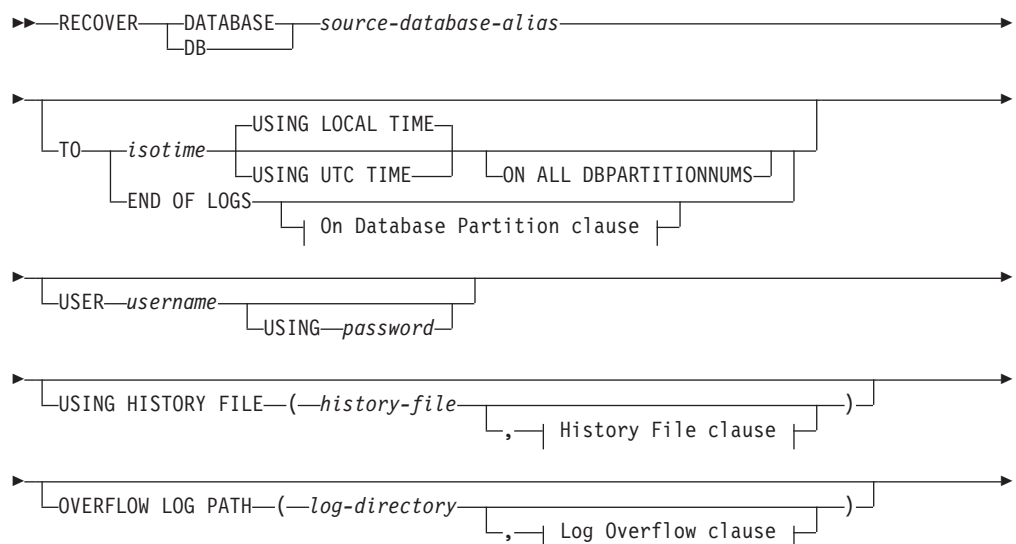
To recover to a new database, one of the following:

- `sysadm`
- `sysctrl`

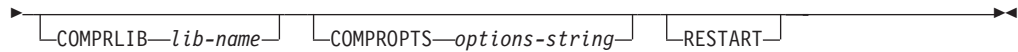
### Required connection:

To recover an existing database, a database connection is required. This command automatically establishes a connection to the specified database and will release the connection when the recover operation finishes. To recover to a new database, an instance attachment and a database connection are required. The instance attachment is required to create the database.

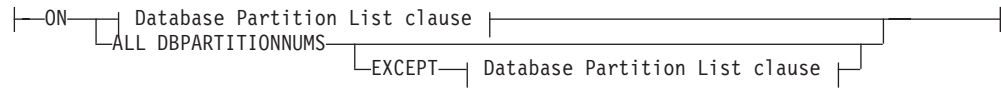
### Command syntax:



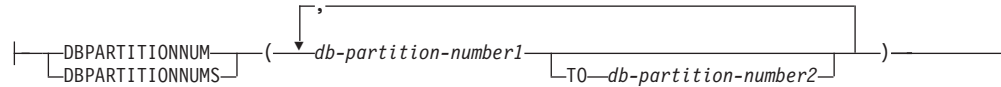
## RECOVER DATABASE



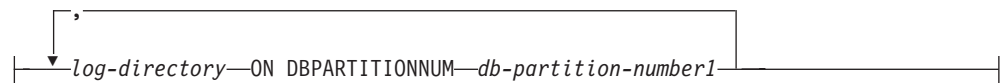
### On Database Partition clause:



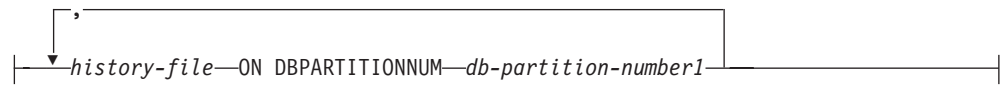
### Database Partition List clause:



### Log Overflow clause:



### History File clause:



### Command parameters:

#### **DATABASE** *database-alias*

The alias of the database that is to be recovered.

#### **USER** *username*

The user name under which the database is to be recovered.

#### **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 recovered (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.nnnnnnn* (year, month, day, hour, minutes, seconds, microseconds), expressed in Coordinated Universal Time (UTC, formerly known as GMT). UTC helps to avoid having the same time stamp associated with different logs (because of a change in time associated with daylight savings time, for example). The time stamp in a backup image is based on the local time at which the backup operation started. The **CURRENT TIMEZONE** special register specifies the difference between UTC and local time at the application server. The difference is represented by a time duration (a decimal number in which the first two digits represent

the number of hours, the next two digits represent the number of minutes, and the last two digits represent the number of seconds). Subtracting CURRENT TIMEZONE from a local time converts that local time to UTC.

#### USING LOCAL TIME

Specifies the point in time to which to recover. This option allows the user to recover to a point in time that is the server's local time rather than UTC time.

##### Notes:

1. If the user specifies a local time for recovery, all messages returned to the user will also be in local time. All times are converted on the server, and in partitioned database environments, on the catalog database partition.
2. The timestamp string is converted to UTC on the server, so the time is local to the server's time zone, not the client's. If the client is in one time zone and the server in another, the server's local time should be used. This is different from the local time option from the Control Center, which is local to the client.
3. If the timestamp string is close to the time change of the clock due to daylight saving time, it is important to know if the stop time is before or after the clock change, and specify it correctly.

#### USING UTC TIME

Specifies the point in time to which to recover.

#### 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.

#### ON ALL DBPARTITIONNUMS

Specifies that transactions are to be rolled forward on all database partitions specified in the *db2nodes.cfg* file. This is the default if a database partition clause is not specified.

#### EXCEPT

Specifies that transactions are to be rolled forward on all database partitions specified in the *db2nodes.cfg* file, except those specified in the database partition list.

#### ON DBPARTITIONNUM / ON DBPARTITIONNUMS

Roll the database forward on a set of database partitions.

*db-partition-number1*

Specifies a database partition number in the database partition list.

*db-partition-number2*

Specifies the second database partition number, so that all database partitions from *db-partition-number1* up to and including *db-partition-number2* are included in the database partition list.

#### USING HISTORY FILE *history-file*

*history-file* ON DBPARTITIONNUM

In a partitioned database environment, allows a different history file

#### OVERFLOW LOG PATH *log-directory*

Specifies an alternate log path to be searched for archived logs during recovery. Use this parameter if log files were moved to a location other

## RECOVER DATABASE

than that specified by the *logpath* database configuration parameter. In a partitioned database environment, this is the (fully qualified) default overflow log path *for all database partitions*. A relative overflow log path can be specified for single-partition databases.

The OVERFLOW LOG PATH command parameter will overwrite the value (if any) of the database configuration parameter *overflowlogpath*.

### COMPRLIB *lib-name*

Indicates the name of the library to be used to perform the decompression. The name must be a fully qualified path referring to a file on the server. If this parameter is not specified, DB2 will attempt to use the library stored in the image. If the backup was not compressed, the value of this parameter will be ignored. If the specified library cannot be loaded, the restore operation will fail.

### COMPROPTS *options-string*

Describes a block of binary data that is passed to the initialization routine in the decompression library. The DB2 database system passes this string directly from the client to the server, so any issues of byte reversal or code page conversion are handled by the decompression library. If the first character of the data block is "@", the remainder of the data is interpreted by the DB2 database system as the name of a file residing on the server. The DB2 database system will then replace the contents of *string* with the contents of this file and pass the new value to the initialization routine instead. The maximum length for the string is 1 024 bytes.

### RESTART

The RESTART keyword can be used if a prior RECOVER operation was interrupted or otherwise did not complete. Starting in v91, a subsequent **RECOVER command** will attempt to continue the previous **RECOVER**, if possible. Using the RESTART keyword forces **RECOVER** to start with a fresh restore and then rollforward to the PIT specified.

### *log-directory* ON DBPARTITIONNUM

In a partitioned database environment, allows a different log path to override the default overflow log path for a specific database partition.

### Examples:

In a single-partition database environment, where the database being recovered currently exists, and so the most recent version of the history file is available in the *dftdbpath*:

1. To use the latest backup image and rollforward to the end of logs using all default values:  

```
RECOVER DB SAMPLE
```
2. To recover the database to a PIT, issue the following. The most recent image that can be used will be restored, and logs applied until the PIT is reached.  

```
RECOVER DB SAMPLE TO 2001-12-31-04.00.00
```
3. To recover the database using a saved version of the history file, issue the following. For example, if the user needs to recover to an extremely old PIT which is no longer contained in the current history file, the user will have to provide a version of the history file from this time period. If the user has saved a history file from this time period, this version can be used to drive the recover.  

```
RECOVER DB SAMPLE TO 1999-12-31-04.00.00
USING HISTORY FILE (/home/user/old1999files/db2rhist.asc)
```



In a single-partition database environment, where the database being recovered does not exist, you must use the USING HISTORY FILE clause to point to a history file.

1. If you have not made any backups of the history file, so that the only version available is the copy in the backup image, the recommendation is to issue a RESTORE followed by a ROLLFORWARD. However, to use RECOVER, you would first have to extract the history file from the image to some location, for example /home/user/oldfiles/db2rhist.asc, and then issue this command. (This version of the history file does not contain any information about log files that are required for rollforward, so this history file is not useful for RECOVER.)

```
RECOVER DB SAMPLE TO END OF LOGS
 USING HISTORY FILE (/home/user/fromimage/db2rhist.asc)
```

2. If you have been making periodic or frequent backup copies of the history, the USING HISTORY clause should be used to point to this version of the history file. If the file is /home/user/myfiles/db2rhist.asc, issue the command:

```
RECOVER DB SAMPLE TO PIT
 USING HISTORY FILE (/home/user/myfiles/db2rhist.asc)
```

(In this case, you can use any copy of the history file, not necessarily the latest, as long as it contains a backup taken before the point-in-time (PIT) requested.)

In a partitioned database environment, where the database exists on all database partitions, and the latest history file is available on *dftdbpath* on all database partitions:

1. To recover the database to a PIT on all nodes. DB2 will verify that the PIT is reachable on all nodes before starting any restore operations.

```
RECOVER DB SAMPLE TO 2001-12-31-04.00.00
```

2. To recover the database to this PIT on all nodes. DB2 will verify that the PIT is reachable on all nodes before starting any restore operations. The RECOVER operation on each node is identical to a single-partition RECOVER.

```
RECOVER DB SAMPLE TO END OF LOGS
```

3. Even though the most recent version of the history file is in the *dftdbpath*, you might want to use several specific history files. Unless otherwise specified, each database partition will use the history file found locally at /home/user/oldfiles/db2rhist.asc. The exceptions are nodes 2 and 4. Node 2 will use: /home/user/node2files/db2rhist.asc, and node 4 will use: /home/user/node4files/db2rhist.asc.

```
RECOVER DB SAMPLE TO 1999-12-31-04.00.00
 USING HISTORY FILE (/home/user/oldfiles/db2rhist.asc,
 /home/user/node2files/db2rhist.asc ON DBPARTITIONNUM 2,
 /home/user/node4files/db2rhist.asc ON DBPARTITIONNUM 4)
```

4. It is possible to recover a subset of nodes instead of all nodes, however a PIT RECOVER can not be done in this case, the recover must be done to EOL.

```
RECOVER DB SAMPLE TO END OF LOGS ON DBPARTITIONNUMS(2 TO 4, 7, 9)
```

In a partitioned database environment, where the database does not exist:

1. If you have not made any backups of the history file, so that the only version available is the copy in the backup image, the recommendation is to issue a RESTORE followed by a ROLLFORWARD. However, to use RECOVER, you would first have to extract the history file from the image to some location, for example, /home/user/oldfiles/db2rhist.asc, and then issue this command.

## RECOVER DATABASE

(This version of the history file does not contain any information about log files that are required for rollforward, so this history file is not useful for the recover.)

```
RECOVER DB SAMPLE TO PIT
USING HISTORY FILE (/home/user/fromimage/db2rhist.asc)
```

2. If you have been making periodic or frequent backup copies of the history, the USING HISTORY clause should be used to point to this version of the history file. If the file is /home/user/myfiles/db2rhist.asc, you can issue the following command:

```
RECOVER DB SAMPLE TO END OF LOGS
USING HISTORY FILE (/home/user/myfiles/db2rhist.asc)
```

### Usage notes:

- Recovering a database might 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.
- If there is a failure during the restore portion of the recover operation, you can reissue the RECOVER DATABASE command. If the restore operation was successful, but there was an error during the rollforward operation, you can issue a ROLLFORWARD DATABASE command, since it is not necessary (and it is time-consuming) to redo the entire recover operation.
- In a partitioned database environment, if there is an error during the restore portion of the recover operation, it is possible that it is only an error on a single database partition. Instead of reissuing the RECOVER DATABASE command, which restores the database on all database partitions, it is more efficient to issue a RESTORE DATABASE command for the database partition that failed, followed by a ROLLFORWARD DATABASE command.

### Related concepts:

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

### Related tasks:

- “Using recover” in *Data Recovery and High Availability Guide and Reference*

## REDISTRIBUTE DATABASE PARTITION GROUP

Redistributes data across the database partitions in a database partition group. 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 database partition. Use the LIST DATABASE DIRECTORY command to determine which database partition is the catalog database partition for each database.

### Scope:

This command affects all database partitions in the database partition group.

### Authorization:

One of the following:

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

### Command syntax:

```

▶▶—REDISTRIBUTE DATABASE PARTITION GROUP—database partition group————▶▶
|
|—UNIFORM————▶▶
|—USING DISTFILE—distfile————▶▶
|—USING TARGETMAP—targetmap————▶▶
|—CONTINUE————▶▶
|—ROLLBACK————▶▶

```

### Command parameters:

#### DATABASE PARTITION GROUP *database partition group*

The name of the database partition group. This one-part name identifies a database partition group described in the SYSCAT.DBPARTITIONGROUPS catalog table. The database partition group cannot currently be undergoing redistribution. Tables in the IBMCATGROUP and the IBMTEMPGROUP database partition groups 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 database partition. After redistribution, all database partitions in the database partition group have approximately the same number of hash partitions.

#### USING DISTFILE *distfile*

If the distribution of distribution key values is skewed, use this option to achieve a uniform redistribution of data across the database partitions of a database partition group.

Use the *distfile* to indicate the current distribution of data across the 4 096 hash partitions.

## REDISTRIBUTE DATABASE PARTITION GROUP

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 distribution map that it uses to redistribute the data across the database partitions in the database partition group as uniformly as possible. After the redistribution, the weight of each database partition in the database partition group is approximately the same (the weight of a database partition is the sum of the weights of all database partitions that map to that database partition).

For example, the input distribution file might 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 distribution map. Data redistribution is done according to this file. If the path is not specified, the current directory is used.

If a database partition included in the target map is not in the database partition group, an error is returned. Issue ALTER DATABASE PARTITION GROUP ADD DBPARTITIONNUM before running REDISTRIBUTE DATABASE PARTITION GROUP.

If a database partition excluded from the target map *is* in the database partition group, that database partition will not be included in the partitioning. Such a database partition can be dropped using ALTER DATABASE PARTITION GROUP DROP DBPARTITIONNUM either before or after REDISTRIBUTE DATABASE PARTITION GROUP.

### CONTINUE

Continues a previously failed REDISTRIBUTE DATABASE PARTITION GROUP operation. If none occurred, an error is returned.

### ROLLBACK

Rolls back a previously failed REDISTRIBUTE DATABASE PARTITION GROUP operation. If none occurred, an error is returned.

### Usage notes:

- When a redistribution operation is done, a message file is written to:
  - The /sql/lib/redist directory on UNIX based systems, using the following format for subdirectories and file name: *database-name.database-partition-group-name.timestamp*.
  - The \sql\lib\redist\ directory on Windows operating systems, using the following format for subdirectories and file name: *database-name\first-eight-characters-of-the-database-partition-group-name\date\time*.
- The time stamp value is the time when the command was issued.

## REDISTRIBUTE DATABASE PARTITION GROUP

- This utility performs intermittent COMMITs during processing.
- Use the ALTER DATABASE PARTITION GROUP statement to add database partitions to a database partition group. This statement permits one to define the containers for the table spaces associated with the database partition group.
- DB2 Parallel Edition for AIX Version 1 syntax, with ADD DBPARTITIONNUM and DROP DBPARTITIONNUM options, is supported for users with *sysadm* or *sysctrl* authority. For ADD DBPARTITIONNUM, containers are created like the containers on the lowest node number of the existing nodes within the database partition group.
- 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 database partition group 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 after the redistribute database partition group operation has completed.
- Database partition groups containing replicated summary tables or tables defined with DATA CAPTURE CHANGES cannot be redistributed.
- Redistribution is not allowed if there are user temporary table spaces with existing declared temporary tables in the database partition group.
- Before starting a redistribute operation, ensure there are no tables in the Load Pending state. Table states can be checked by using the **LOAD QUERY** command. If you discover data in the wrong database partition as a result of a redistribute operation, there are two options. You can:
  1. unload the table, drop it and then reload the table, or
  2. use a new target map to redistribute the database partition group again.

### Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODEGROUP can be substituted for DATABASE PARTITION GROUP.

### Related tasks:

- “Redistributing data across database partitions” in *Performance Guide*
- “Altering a database partition group” in *Administration Guide: Implementation*

### Related reference:

- “sqludrdt API - Redistribute data across a database partition group” in *Administrative API Reference*
- “LIST DATABASE DIRECTORY ” on page 523
- “RUNSTATS ” on page 702
- “REBIND ” on page 619
- “REDISTRIBUTE DATABASE PARTITION GROUP command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*

---

## REFRESH LDAP

Refreshes the cache on a local machine with updated information when the information in Lightweight Directory Access Protocol (LDAP) has been changed.

### Authorization:

None

### Required connection:

None

### Command syntax:



### Command parameters:

#### CLI CFG

Specifies that the CLI configuration is to be refreshed. This parameter is not supported on AIX or the Solaris operating system.

#### 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.

The `REFRESH LDAP DB DIR` and `REFRESH LDAP NODE DIR` commands remove the LDAP database or node entries found in the local database or node directories. The database or node entries will be added to the local database or node directories again when the user connects to a database or attaches to an instance found in LDAP, and `DB2LDAPCACHE` is either not set or set to `YES`.

### Related tasks:

- “Refreshing LDAP entries in local database and node directories” in *Administration Guide: Implementation*

## REGISTER

Registers the DB2 server in the network directory server.

### Authorization:

None

### Required connection:

None

### Command syntax:

```

▶▶ REGISTER [DB2 SERVER] [IN] [ADMIN] | LDAP path |

```

### LDAP path:

```

| LDAP [NODE AS] nodename |
▶ PROTOCOL [TCPIP [HOSTNAME—hostname] [SVCENAME—svcename] [SECURITY—SOCKS]
| [TCPIP4 [HOSTNAME—hostname] [SVCENAME—svcename] [SECURITY—SOCKS]
| [TCPIP6 [HOSTNAME—hostname] [SVCENAME—svcename]
| [NPIPE]
▶ [REMOTE—computer] [INSTANCE—instance] [NODETYPE [SERVER] [MPP] [DCS] [OSTYPE—ostype]
▶ [WITH—"comments"] [USER—username] [PASSWORD—password]

```

### Command parameters:

**IN** Specifies the network directory server on which to register the DB2 server. The valid value is: LDAP for an LDAP (Lightweight Directory Access Protocol) directory server.

### ADMIN

Specifies that an administration server node is to be registered.

### NODE/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 can 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, TCPIP4, TCPIP6, and NPIPE. Specify NPIPE to use Windows Named Pipes. NPIPE is only supported on Windows operating systems.

## REGISTER

### **HOSTNAME hostname**

Specifies the TCP/IP host name (or IP address). IP address can be an IPv4 or IPv6 address when using protocol, TCPIP. IP address must be an IPv4 address when using protocol, TCPIP4. IP address must be an IPv6 address when using protocol, TCPIP6.

### **SVCENAME svcename**

Specifies the TCP/IP service name or port number.

### **SECURITY SOCKS**

Specifies that TCP/IP SOCKS is to be used. This parameter is only supported with IPv4. When protocol TCPIP is specified, the underlying protocol used will be IPv4.

### **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 Windows operating systems, this is the computer name. For UNIX based systems, this is the TCP/IP host name.

### **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 Enterprise Server Edition. This is the default.

**MPP** Specify the MPP node type for a DB2 Enterprise Server Edition - Extended (partitioned database) server.

**DCS** Specify the DCS node type when registering a host database server.

### **OSTYPE ostype**

Specifies the operating system type of the server machine. Valid values are: AIX, NT, HPUX, SUN, MVS, OS400, VM, VSE and LINUX. If an operating system type is not specified, the local operating system type will be used for a local server and no operating system type will be used for a remote server.

### **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:**

Register the DB2 server once for each protocol that the server supports.

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.

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.

If the REGISTER command is used to register a local DB2 instance in LDAP, and one or both of NODETYPE and OSTYPE are specified, they will be replaced with the values retrieved from the local system. If the REGISTER command is used to register a remote DB2 instance in LDAP, and one or both of NODETYPE and OSTYPE are not specified, the default value of SERVER and Unknown will be used, respectively.

If the REGISTER command is 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.

**Related concepts:**

- “Lightweight Directory Access Protocol (LDAP) overview” in *Administration Guide: Implementation*

**Related tasks:**

- “Registration of DB2 servers after installation” in *Administration Guide: Implementation*

**Related reference:**

- “db2LdapRegister API - Register the DB2 server on the LDAP server” in *Administrative API Reference*
- “DEREGISTER ” on page 415
- “UPDATE LDAP NODE ” on page 780

## REGISTER XMLSCHEMA

Registers an XML schema with the XML schema repository (XSR).

**Authorization:**

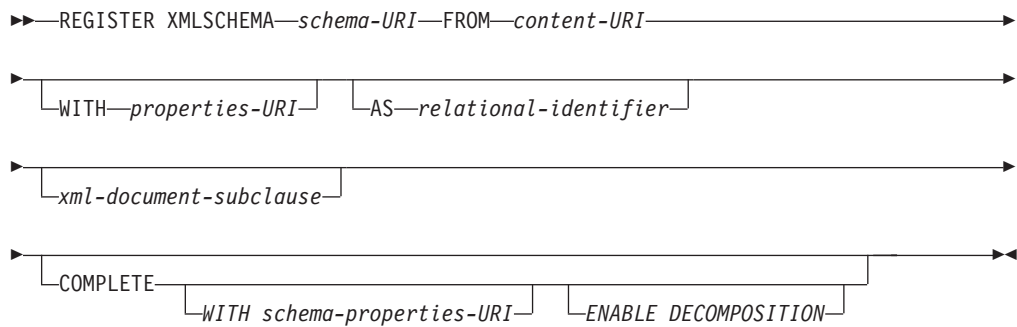
One of the following:

- SYSADM or DBADM
- IMPLICIT\_SCHEMA database authority if the SQL schema does not exist
- CREATEIN privilege if the SQL schema exists

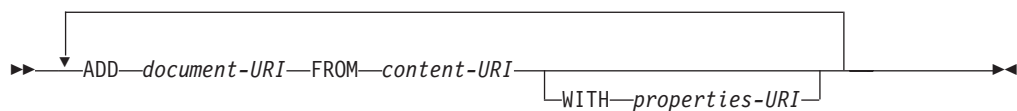
**Required connection:**

Database

**Command syntax:**



*xml-document-subclause:*



**Description:**

*schema-URI*

Specifies the URI, as referenced by XML instance documents, of the XML schema being registered.

**FROM** *content-URI*

Specifies the URI where the XML schema document is located. Only a local file specified by a file scheme URI is supported.

**WITH** *properties-URI*

Specifies the URI of a properties document for the XML schema. Only a local file specified by a file scheme URI is supported.

**AS** *relational-identifier*

Specifies a name that can be used to refer to the XML schema being registered. The relational name can be specified as a two-part SQL identifier, consisting of the SQL schema and the XML schema name, having the following format: SQLschema.name. The default relational schema, as

defined in the CURRENT SCHEMA special register, is used if no schema is specified. If no name is provided, a unique value is generated.

**COMPLETE**

Indicates that there are no more XML schema documents to be added. If specified, the schema is validated and marked as usable if no errors are found.

**WITH** *schema-properties-URI*

Specifies the URI of a properties document for the XML schema. Only a local file specified by a file scheme URI is supported.

**ENABLE DECOMPOSITION**

Specifies that this schema is to be used for decomposing XML documents.

**ADD** *document-URI*

Specifies the URI of an XML schema document to be added to this schema, as the document would be referenced from another XML document.

**FROM** *content-URI*

Specifies the URI where the XML schema document is located. Only a local file specified by a file scheme URI is supported.

**WITH** *properties-URI*

Specifies the URI of a properties document for the XML schema. Only a local file specified by a file scheme URI is supported.

**Example:**

```
REGISTER XMLSCHEMA 'http://myPOschema/PO.xsd'
FROM 'file:///c:/TEMP/PO.xsd'
WITH 'file:///c:/TEMP/schemaProp.xml'
AS user1.POschema
```

**Usage notes:**

- Before an XML schema document can be referenced and be available for validation and annotation, it must first be registered with the XSR. This command performs the first step of the XML schema registration process, by registering the primary XML schema document. The final step of the XML schema registration process requires that the COMPLETE XMLSCHEMA command run successfully for the XML schema. Alternatively, if there are no other XML schema documents to be included, issue the REGISTER XMLSCHEMA command with the COMPLETE keyword to complete registration in one step.
- When registering an XML schema in the database, a larger application heap (APPLHEAPSZ) may be required depending on the size of the XML schema. The recommended size is 1024 but larger schemas will require additional memory.

**Related reference:**

- “ADD XMLSCHEMA DOCUMENT ” on page 339
- “COMPLETE XMLSCHEMA ” on page 394

---

## REGISTER XSROBJECT

Registers an XML object in the database catalogs. Supported objects are DTDs and external entities.

### Authorization:

One of the following:

- SYSADM or DBADM
- IMPLICIT\_SCHEMA database authority if the SQL schema does not exist
- CREATEIN privilege if the SQL schema exists

### Required connection:

Database

### Command syntax:

```

▶▶ REGISTER XSROBJECT system-ID [PUBLIC public-id] FROM content-URI
▶ [AS relational-identifier] [DTD | EXTERNAL ENTITY]

```

### Command parameters:

*system-id*

Specifies the system ID that is specified in the XML object declaration.

**PUBLIC** *public-id*

Specifies an optional PUBLIC ID in the XML object declaration.

**FROM** *content-URI*

Specifies the URI where the content of an XML schema document is located. Only a local file specified by a file scheme URI is supported.

**AS** *relational-identifier*

Specifies a name that can be used to refer to the XML object being registered. The relational name can be specified as a two-part SQL identifier consisting of the relational schema and name separated by a period, for example "JOHNDOE.EMPLOYEEEDTD". If no relational schema is specified, the default relational schema defined in the special register CURRENT\_SCHEMA is used. If no name is specified, one is generated automatically.

**DTD** Specifies that the object being registered is a Data Type Definition document (DTD).

**EXTERNAL ENTITY**

Specifies that the object being registered is an external entity.

### Examples:

1. Given this sample XML document which references an external entity:

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE copyright [
 <!ELEMENT copyright (#PCDATA)>
 <!ENTITY c SYSTEM "http://www.xmlwriter.net/copyright.xml">
]>
<copyright>&c;</copyright>
```

Before this document can be successfully inserted into an XML column, the external entity needs to be registered. The following command registers an entity where the entity content is stored locally in C:\TEMP:

```
REGISTER XSROBJECT 'http://www.xmlwriter.net/copyright.xml'
FROM 'c:\temp\copyright.xml' EXTERNAL ENTITY
```

2. Given this XML document fragment which references a DTD:

```
<!--inform the XML processor
that an external DTD is referenced-->
<?xml version="1.0" standalone="no" ?>

<!--define the location of the
external DTD using a relative URL address-->
<!DOCTYPE document SYSTEM "http://www.xmlwriter.net/subjects.dtd">

<document>
 <title>Subjects available in Mechanical Engineering.</title>
 <subjectID>2.303</subjectID>
 <subjectname>Fluid Mechanics</subjectname>
 ...
```

Before this document can be successfully inserted into an XML column, the DTD needs to be registered. The following command registers a DTD where the DTD definition is stored locally in C:\TEMP and the relational identifier to be associated with the DTD is "TEST.SUBJECTS":

```
REGISTER XSROBJECT 'http://www.xmlwriter.net/subjects.dtd'
FROM 'file:///c:/temp/subjects.dtd' AS TEST.SUBJECTS DTD
```

3. Given this sample XML document which references a public external entity:

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE copyright [
 <!ELEMENT copyright (#PCDATA)>
 <!ENTITY c PUBLIC "-//W3C//TEXT copyright//EN"
 "http://www.w3.org/xmlspec/copyright.xml">
]>
<copyright>&c;</copyright>
```

Before this document can be successfully inserted into an XML column, the public external entity needs to be registered. The following command registers an entity where the entity content is stored locally in C:\TEMP:

```
REGISTER XSROBJECT 'http://www.w3.org/xmlspec/copyright.xml'
PUBLIC "-//W3C//TEXT copyright//EN' FROM 'file:///c:/temp/copyright.xml'
EXTERNAL ENTITY
```

#### Related concepts:

- "XSR object registration" in *XML Guide*
- "XSR objects" in *XML Guide*

#### Related reference:

- "REGISTER XMLSCHEMA " on page 640

## REORG INDEXES/TABLE

Reorganizes an index or a table.

You can reorganize all indexes defined on a table by rebuilding the index data into unfragmented, physically contiguous pages. Alternatively, you have the option of reorganizing specific indexes on a range partitioned table.

If you specify the CLEANUP ONLY option of the index clause, cleanup is performed without rebuilding the indexes. This command cannot be used against indexes on declared temporary tables (SQLSTATE 42995).

The table option reorganizes a table by reconstructing the rows to eliminate fragmented data, and by compacting information.

### Scope:

This command affects all database partitions in the database partition group.

### Authorization:

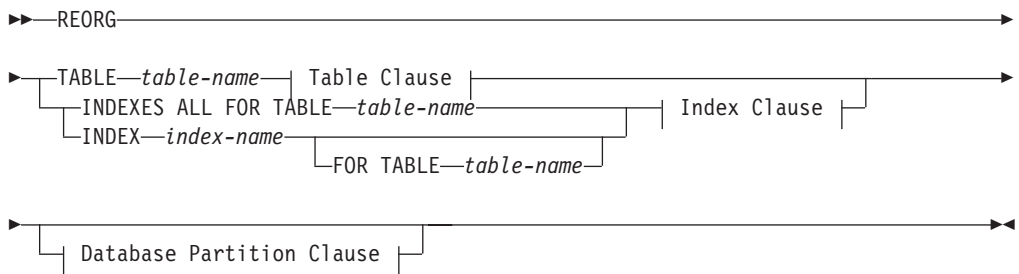
One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the table.

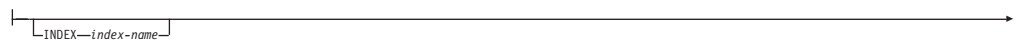
### Required connection:

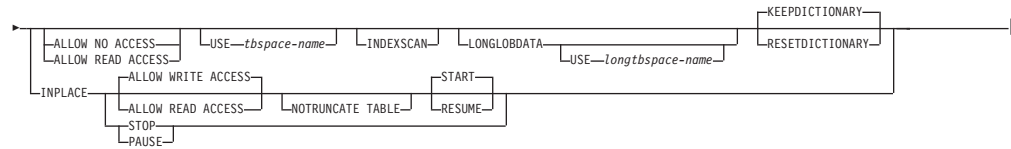
Database

### Command syntax:

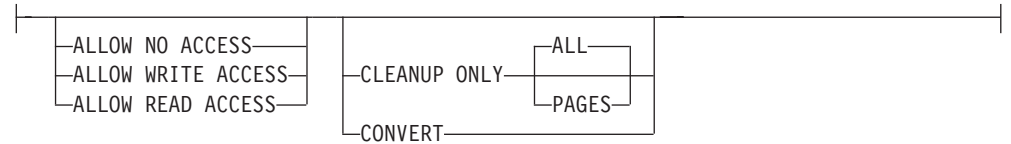


### Table Clause:

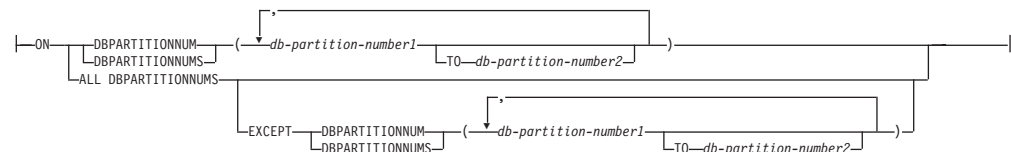




**Index Clause:**



**Database Partition Clause:**



**Command parameters:**

**INDEXES ALL FOR TABLE table-name**

Specifies the table whose indexes are to be reorganized. The table can be in a local or a remote database.

**INDEX index-name**

Specifies an individual index to be reorganized on a partitioned table. Reorganization of individual indexes are ONLY supported for non-partitioned indexes on a partitioned table. This parameter is not supported for block indexes.

**FOR TABLE table-name**

Specifies the table name location of the individual index being reorganized on a partitioned table. This parameter is optional, given that index names are unique across the database.

**ALLOW NO ACCESS**

Specifies that no other users can access the table while the indexes are being reorganized.

**ALLOW READ ACCESS**

Specifies that other users can have read-only access to the table while the indexes are being reorganized. This access level is not supported for REORG INDEXES of a partitioned table unless the CLEANUP ONLY option is specified.

**ALLOW WRITE ACCESS**

Specifies that other users can read from and write to the table while the indexes are being reorganized. This access level is not supported for multi-dimensionally clustered (MDC) tables, partitioned tables, extended indexes, or tables containing a column with the XML data type unless the CLEANUP ONLY option is specified.

When no ACCESS mode is specified, one will be chosen for you in the following way:

## REORG INDEXES/TABLE

Table 10. Default table access chosen based on the command, table type and additional parameters specified for the index clause:

Command	Table type	Additional parameters specified for index clause	Default access mode
REORG INDEXES	non-partitioned table	any	ALLOW READ ACCESS
REORG INDEXES	partitioned table	none specified	ALLOW NO ACCESS
REORG INDEXES	partitioned table	CLEANUP ONLY specified	ALLOW READ ACCESS
REORG INDEX	partitioned table	any	ALLOW READ ACCESS

### CLEANUP ONLY

When CLEANUP ONLY is requested, a cleanup rather than a full reorganization will be done. The indexes will not be rebuilt and any pages freed up will be available for reuse by indexes defined on this table only.

The CLEANUP ONLY PAGES option will search for and free committed pseudo empty pages. A committed pseudo empty page is one where all the keys on the page are marked as deleted and all these deletions are known to be committed. The number of pseudo empty pages in an indexes can be determined by running runstats and looking at the NUM EMPTY LEAFS column in SYSCAT.INDEXES. The PAGES option will clean the NUM EMPTY LEAFS if they are determined to be committed.

The CLEANUP ONLY ALL option will free committed pseudo empty pages, as well as remove committed pseudo deleted keys from pages that are not pseudo empty. This option will also try to merge adjacent leaf pages if doing so will result in a merged leaf page that has at least PCTFREE free space on the merged leaf page, where PCTFREE is the percent free space defined for the index at index creation time. The default PCTFREE is ten percent. If two pages can be merged, one of the pages will be freed. The number of pseudo deleted keys in an index, excluding those on pseudo empty pages, can be determined by running runstats and then selecting the NUMRIDS DELETED from SYSCAT.INDEXES. The ALL option will clean the NUMRIDS DELETED and the NUM EMPTY LEAFS if they are determined to be committed.

**ALL** Specifies that indexes should be cleaned up by removing committed pseudo deleted keys and committed pseudo empty pages.

### PAGES

Specifies that committed pseudo empty pages should be removed from the index tree. This will not clean up pseudo deleted keys on pages that are not pseudo empty. Since it is only checking the pseudo empty leaf pages, it is considerably faster than using the ALL option in most cases.

### CONVERT

If you are not sure whether the table you are operating on has a type-1 or type-2 index, but want type-2 indexes, you can use the CONVERT option. If the index is type 1, this option will convert it into type 2. If the index is already type 2, this option has no effect.

All indexes created by DB2 prior to Version 8 are type-1 indexes. All indexes created by Version 8 are Type 2 indexes, except when



you create an index on a table that already has a type 1 index. In this case the new index will also be of type 1.

Using the INSPECT command to determine the index type can be slow. CONVERT allows you to ensure that the new index will be Type 2 without your needing to determine its original type.

Use the ALLOW READ ACCESS or ALLOW WRITE ACCESS option to allow other transactions either read-only or read-write access to the table while the indexes are being reorganized. While ALLOW READ ACCESS and ALLOW WRITE ACCESS allow access to the table, during the period in which the reorganized copies of the indexes are made available, no access to the table is allowed.

**TABLE table-name**

Specifies the table to reorganize. The table can be in a local or a remote database. The name or alias in the form: *schema.table-name* can be used. The *schema* is the user name under which the table was created. If you omit the schema name, the default schema is assumed.

For typed tables, the specified table name must be the name of the hierarchy's root table.

You cannot specify an index for the reorganization of a multidimensional clustering (MDC) table. In place reorganization of tables cannot be used for MDC tables.

**INDEX index-name**

Specifies the index to use when reorganizing the table. If you do not specify the fully qualified name in the form: *schema.index-name*, the default schema is assumed. 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.

For an in place table reorganization, if a clustering index is defined on the table and an index is specified, it must be clustering index. If the in place option is not specified, any index specified will be used. If you do not specify the name of an index, the records are reorganized without regard to order. If the table has a clustering index defined, however, and no index is specified, then the clustering index is used to cluster the table. You cannot specify an index if you are reorganizing an MDC table.

**ALLOW NO ACCESS**

Specifies that no other users can access the table while the table is being reorganized. When reorganizing a partitioned table, this is the default. Reorganization of a partitioned table occurs offline.

**ALLOW READ ACCESS**

Allow only read access to the table during reorganization. This is the default for a non-partitioned table.

**INPLACE**

Reorganizes the table while permitting user access.

In place table reorganization is allowed only on non-partitioned and non-MDC tables with type-2 indexes, but without extended indexes and with no indexes defined over XML columns in the table. In place table reorganization takes place asynchronously, and might not be effective immediately.

### **ALLOW READ ACCESS**

Allow only read access to the table during reorganization.

### **ALLOW WRITE ACCESS**

Allow write access to the table during reorganization. This is the default behavior.

### **NOTRUNCATE TABLE**

Do not truncate the table after in place reorganization. During truncation, the table is S-locked.

### **START**

Start the in place REORG processing. Because this is the default, this keyword is optional.

**STOP** Stop the in place REORG processing at its current point.

### **PAUSE**

Suspend or pause in place REORG for the time being.

### **RESUME**

Continue or resume a previously paused in place table reorganization. When an online reorganization is resumed and you want the same options as when the reorganization was paused, you must specify those options again while resuming.

### **USE tbspace-name**

Specifies the name of a system temporary table space in which to store a temporary copy of the table being reorganized. If you do not provide a table space name, the database manager stores a working copy of the table in the table spaces that contain the table being reorganized.

For an 8KB, 16KB, or 32KB table object, if the page size of the system temporary table space that you specify does not match the page size of the table spaces in which the table data resides, the DB2 database product will try to find a temporary table space of the correct size of the LONG/LOB objects. Such a table space must exist for the reorganization to succeed.

When you have two temporary tablespaces of the same page size, and you specify one of them in the USE clause, they will be used in a round robin fashion if there is an index in the table being reorganized. Say you have two tablespaces, `tempspace1` and `tempspace2`, both of the same page size and you specify `tempspace1` in the **REORG** command with the USE option. When you perform **REORG** the first time, `tempspace1` is used. The second time, `tempspace2` is used. The third time, `tempspace1` is used and so on. To avoid this, you should drop one of the temporary tablespaces.

For partitioned tables, the table space is used as temporary storage for the reorganization of all the data partitions in the table. Reorganization of a partitioned table reorganizes a single data partition at a time. The amount of space required is equal to the largest data partition in the table, and not the entire table.

If you do not supply a table space name for a partitioned table, the table space where each data partition is located is used for

temporary storage of that data partition. There must be enough free space in each data partition's table space to hold a copy of the data partition.

**INDEXSCAN**

For a clustering REORG an index scan will be used to re-order table records. Reorganize table rows by accessing the table through an index. The default method is to scan the table and sort the result to reorganize the table, using temporary table spaces as necessary. Even though the index keys are in sort order, scanning and sorting is typically faster than fetching rows by first reading the row identifier from an index.

**LONGLOBDATA**

Long field and LOB data are to be reorganized.

This is not required even if the table contains long or LOB columns. The default is to avoid reorganizing these objects because it is time consuming and does not improve clustering.

**USE longtblspace-name**

This is an optional parameter, which can be used to specify the name of a temporary tablespace to be used for rebuilding long data. If no temporary tablespace is specified for either the table object or for the long objects, the objects will be constructed in the tablespace they currently reside. If a temporary tablespace is specified for the table but this parameter is not specified, then the tablespace used for base reorg data will be used, unless the page sizes differ. In this situation, the DB2 database system will attempt to choose a temporary container of the appropriate page size to create the long objects in.

If USE-longtblspace is specified, USE-tblspace must also be specified. If it is not, the longtblspace argument is ignored.

**KEEPDICTIONARY**

If the COMPRESS attribute for the table is YES and the table has a compression dictionary then no new dictionary is built. All the rows processed during reorganization are subject to compression using the existing dictionary. If the COMPRESS attribute for the table is NO and the table has a compression dictionary then reorg processing will remove the dictionary and all the rows in the newly reorganized table will be in non-compressed format. It is not possible to compress long, LOB, index, or XML objects.

*Table 11. REORG KEEPDICTIONARY*

Compress	Dictionary Exists	Result; outcome	
Y	Y	Preserve dictionary; rows compressed	
Y	N	Build dictionary; rows compressed	
N	Y	Preserve dictionary; all rows uncompressed	
N	N	No effect; all rows uncompressed	

For any reinitialization or truncation of a table (such as for a replace operation), if the compress attribute for the table is NO, the dictionary is discarded if one exists. Conversely, if a dictionary exists and the compress attribute for the table is YES then a truncation will save the dictionary and not discard it. The dictionary is logged in its entirety for recovery purposes and for future support with data capture changes (i.e. replication).

**RESETDICTIONARY**

If the COMPRESS attribute for the table is YES then a new row compression dictionary is built. All the rows processed during reorganization are subject to compression using this new dictionary. This dictionary replaces any previous dictionary. If the COMPRESS attribute for the table is NO and the table does have an existing compression dictionary then reorg processing will remove the dictionary and all rows in the newly reorganized table will be in non-compressed format. It is not possible to compress long, LOB, index, or XML objects.

Table 12. REORG RESETDICTIONARY

Compress	Dictionary Exists	Result; outcome	
Y	Y	Build new dictionary*; rows compressed	
Y	N	Build new dictionary; rows compressed	
N	Y	Remove dictionary; all rows uncompressed	
N	N	No effect; all rows uncompressed	

\* - If a dictionary exists and the compression attribute is enabled but there is currently insufficient data in the table to build a new dictionary, the RESETDICTIONARY operation will keep the existing dictionary. Rows which are smaller in size than the internal minimum record length and rows which do not demonstrate a savings in record length when an attempt is made to compress them are considered 'insufficient' in this case.

**Examples:**

To reorganize a table to reclaim space and use the temporary table space mytemp1, enter the following command:

```
db2 reorg table homer.employee use mytemp1
```

To reorganize tables in a partitiongroup consisting of nodes 1, 2, 3, and 4 of a four-node system, you can enter either of the following commands:

```
db2 reorg table employee index empid on dbpartitionnum (1,3,4)
```

```
db2 reorg table homer.employee index homer.empid on all
dbpartitionnums except dbpartitionnum (2)
```

To clean up the pseudo deleted keys and pseudo empty pages in all the indexes on the EMPLOYEE table while allowing other transactions to read and update the table, enter:

```
db2 reorg indexes all for table homer.employee allow write
access cleanup only
```

To clean up the pseudo empty pages in all the indexes on the EMPLOYEE table while allowing other transactions to read and update the table, enter:

```
db2 reorg indexes all for table homer.employee allow write
access cleanup only pages
```

To reorganize the EMPLOYEE table using the system temporary table space TEMPSPACE1 as a work area, enter:

```
db2 reorg table homer.employee use temp space1
```

To start, pause, and resume an in place reorg of the EMPLOYEE table with the default schema HOMER, which is specified explicitly in previous examples, enter the following commands:

```
db2 reorg table employee index empid inplace start
db2 reorg table employee inplace pause
db2 reorg table homer.employee inplace allow read access
nottruncate table resume
```

The command to resume the reorg contains additional keywords to specify read access only and to skip the truncation step, which share-locks the table.

#### Usage notes:

Restrictions:

- The **REORG** utility does not support the use of nicknames.
- The **REORG TABLE** command is not supported for declared temporary tables.
- The **REORG TABLE** command cannot be used on views.
- Reorganization of a table is not compatible with range-clustered tables, because the range area of the table always remains clustered.
- **REORG TABLE** cannot be used on a partitioned table in a DMS table space while an online backup of ANY table space in which the table resides, including LOBs and indexes, is being performed.
- **REORG TABLE** cannot use an index that is based on an index extension.
- If a table is in reorg pending state, an inplace reorg is not allowed on the table.
- For partitioned tables:
  - REORG is supported at the table level. Reorganization of an individual data partition can be achieved by detaching the data partition, reorganizing the resulting non-partitioned table and then re-attaching the data partition.
  - The table must have an ACCESS\_MODE in SYSCAT.TABLES of Full Access.
  - Reorganization skips data partitions that are in a restricted state due to an attach or detach operation
  - If an error occurs, the non-partitioned indexes of the table are marked bad, and are rebuilt on the next access to the table.
  - If a reorganization operation fails, some data partitions may be in a reorganized state and others may not. When the REORG TABLE command is reissued, all the data partitions will be reorganized regardless of the data partition's reorganization state.

## REORG INDEXES/TABLE

- When reorganizing indexes on partitioned tables, it is recommended that you perform a runstats operation after asynchronous index cleanup is complete in order to generate accurate index statistics in the presence of detached data partitions. To determine whether or not there are detached data partitions in the table, you can check the status field in SYSDATAPARTITIONS and look for the value "I" (index cleanup) or "D" (detached with dependant MQT).

Information about the current progress of table reorganization is written to the history file for database activity. The history file contains a record for each reorganization event. To view this file, execute the LIST HISTORY command for the database that contains the table you are reorganizing.

You can also use table snapshots to monitor the progress of table reorganization. Table reorganization monitoring data is recorded regardless of the Database Monitor Table Switch setting.

If an error occurs, an SQLCA dump is written to the history file. For an in-place table reorganization, the status is recorded as PAUSED.

When an indexed table has been modified many times, the data in the indexes might become fragmented. If the table is clustered with respect to an index, the table and index can get out of cluster order. Both of these factors can adversely affect the performance of scans using the index, and can impact the effectiveness of index page prefetching. REORG INDEX or REORG INDEXES can be used to reorganize one or all of the indexes on a table. Index reorganization will remove any fragmentation and restore physical clustering to the leaf pages. Use REORGCHK to help determine if an index needs reorganizing. Be sure to complete all database operations and release all locks before invoking index reorganization. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.

Indexes might not be optimal following an in-place REORG TABLE operation, since only the data object and not the indexes are reorganized. It is recommended that you perform a REORG INDEXES after an in place REORG TABLE operation. Indexes are completely rebuilt during the last phase of a classic REORG TABLE, however, so reorganizing indexes is not necessary.

Tables that have been modified so many times that data is fragmented and access performance is noticeably slow are candidates for the REORG TABLE command. You should also invoke this utility after altering the inline length of a structured type column in order to benefit from the altered inline length. Use REORGCHK to determine whether a table needs reorganizing. Be sure to complete all database operations and release all locks before invoking REORG TABLE. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK. After reorganizing a table, use RUNSTATS to update the table statistics, and REBIND to rebind the packages that use this table. The reorganize utility will implicitly close all the cursors.

If the table contains mixed row format because the table value compression has been activated or deactivated, an offline table reorganization can convert all the existing rows into the target row format.

If the table is distributed across several database partitions, and the table or index reorganization fails on any of the affected database partitions, only the failing database partitions will have the table or index reorganization rolled back.

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.

To complete a table space roll-forward recovery following a table reorganization, both regular and large table spaces must be enabled for roll-forward recovery.

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 or DMS).

### Related concepts:

- “Data row compression” in *Administration Guide: Implementation*
- “Table reorganization” in *Performance Guide*

### Related reference:

- “db2Reorg API - Reorganize an index or a table” in *Administrative API Reference*
- “GET SNAPSHOT ” on page 487
- “REBIND ” on page 619
- “REORGCHK ” on page 654
- “REORG INDEXES/TABLE command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “SNAPTAB\_REORG administrative view and SNAP\_GET\_TAB\_REORG table function – Retrieve table reorganization snapshot information” in *Administrative SQL Routines and Views*
- “REORGCHK\_TB\_STATS procedure – Retrieve table statistics for reorganization evaluation” in *Administrative SQL Routines and Views*
- “REORGCHK\_IX\_STATS procedure – Retrieve index statistics for reorganization evaluation” in *Administrative SQL Routines and Views*

## REORGCHK

Calculates statistics on the database to determine if tables or indexes, or both, need to be reorganized or cleaned up.

**Scope:**

This command can be issued from any database partition in the `db2nodes.cfg` file. It can be used to update table and index statistics in the catalogs.

**Authorization:**

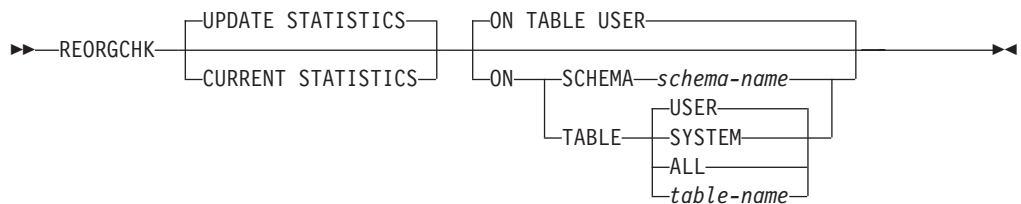
One of the following:

- *sysadm* or *dbadm* authority
- CONTROL privilege on the table.

**Required connection:**

Database

**Command syntax:**



**Command parameters:**

**UPDATE STATISTICS**

Calls the RUNSTATS routine to update table and index statistics, and then uses the updated statistics to determine if table or index reorganization is required.

If a portion of the table resides on the database partition where REORGCHK has been issued, the utility executes on this database partition. If the table does not exist on this database partition, the request is sent to the first database partition in the database partition group that holds a portion of the table. RUNSTATS then executes on this database partition.

**CURRENT STATISTICS**

Uses the current table statistics to determine if table reorganization is required.

**ON SCHEMA schema-name**

Checks all the tables created under the specified schema.

**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. For typed tables, the specified table name must be the name of the hierarchy's root table.

**Examples:**

Issue the following command against the SAMPLE database:

```
db2 reorgchk update statistics on table system
```

In the resulting output, the terms for the table statistics (formulas 1-3) mean:

**CARD**

(CARDINALITY) 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.

**ACTBLK**

Total number of active blocks for a multidimensional clustering (MDC) table. This field is only applicable to tables defined using the ORGANIZE BY clause. It indicates the number of blocks of the table that contain data.

**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.

**NPARTITIONS**

Number of partitions if this is a partitioned table, otherwise 1.

**F1** Results of Formula 1.

**F2** Results of Formula 2.

**F3** Results of Formula 3. This formula indicates the amount of space that is wasted in a table. This is measured in terms of the number of empty pages and the number of pages that include data that exists in the pages of a table. In multi-dimensional clustering (MDC) tables, the number of empty blocks and the number of blocks that include data is measured.

**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.

The table name is truncated to 30 characters, and the ">" symbol in the thirty-first column represents the truncated portion of the table name. An "\*" suffix to a table name indicates it is an MDC table. An "\*" suffix to an index name indicates it is an MDC dimension index.

The terms for the index statistics (formulas 4-8) mean:

**INDCARD**

(INDEX CARDINALITY) Number of index entries in the index. This could be different than table cardinality for some indexes. For example, for indexes on XML columns the index cardinality is likely greater than the table cardinality.

**LEAF** Total number of index leaf pages (NLEAF).

**ELEAF**

Number of pseudo empty index leaf pages (NUM\_EMPTY\_LEAFS)

A pseudo empty index leaf page is a page on which all the RIDs are marked as deleted, but have not been physically removed.

**NDEL** Number of pseudo deleted RIDs (NUMRIDS\_DELETED)

A pseudo deleted RID is a RID that is marked deleted. This statistic reports pseudo deleted RIDs on leaf pages that are not pseudo empty. It does not include RIDs marked as deleted on leaf pages where all the RIDs are marked deleted.

**KEYS** Number of unique index entries that are not marked deleted (FULLKEYCARD)

**LEAF\_RECSIZE**

Record size of the index entry on a leaf page. This is the average size of the index entry excluding any overhead and is calculated from the average column length of all columns participating in the index.

**NLEAF\_RECSIZE**

Record size of the index entry on a non-leaf page. This is the average size of the index entry excluding any overhead and is calculated from the average column length of all columns participating in the index except any INCLUDE columns.

**LEAF\_PAGE\_OVERHEAD**

Reserved space on the index leaf page for internal use.

**NLEAF\_PAGE\_OVERHEAD**

Reserved space on the index non-leaf page for internal use.

**INDEXPAGESIZE**

Page size of the table space in which the index resides, specified at the time of index or table creation. If not specified, INDEXPAGESIZE has the same value as TABLEPAGESIZE.

**LVLS** Number of index levels (NLEVELS)

**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.

**LEAF\_RECSIZE\_OVERHEAD**

Index record overhead on a leaf page. For indexes on tables in LARGE table spaces the overhead is 11 for partitioned tables and 9 for other tables. For indexes on tables in REGULAR table spaces these values are 9 for partitioned tables and 7 for others. The only exception to these rules are XML paths and XML regions indexes where the overhead is always 9. This information is also available in the table below for easy reference.

**NLEAF\_RECSIZE\_OVERHEAD**

Index record overhead on a non-leaf page. For indexes on tables in LARGE table spaces the overhead is 14 for partitioned tables and 12 for other tables. For indexes on tables in REGULAR table spaces these values are 12 for partitioned tables and 10 for others. The only exception to these rules are XML paths and XML regions indexes where the overhead is always 12. This information is also available in the table below for easy reference.

**DUPKEYSIZE**

Size of duplicate keys on index leaf pages. For indexes on tables in LARGE table spaces the DUPKEYSIZE is 9 for partitioned tables and 7 for other tables. For indexes on tables in REGULAR table spaces these values are 7 for partitioned tables and 5 for others. The only exception to these rules are XML paths and XML regions indexes where the DUPKEYSIZE is always 7. This information is also available in the table below for easy reference.

*Table 13. LEAF\_RECSIZE\_OVERHEAD, NLEAF\_RECSIZE\_OVERHEAD, and DUPKEYSIZE values are a function of index type, table partitioning, and table space type*

Variable	Data in REGULAR table space			Data in LARGE table space**		
	Regular Table		Partitioned Table	Regular Table		Partitioned Table
	XML paths or regions index	All other indexes	All indexes	XML paths or regions index	All other indexes	All indexes
LEAF_RECSIZE_OVERHEAD	9	7	9	9	9	11
NLEAF_RECSIZE_OVERHEAD	12	10	12	12	12	14
DUPKEYSIZE	7	5	7	7	7	9

\*\* For indexes on tables in large table spaces the indexes will be assumed to have large RIDs. This may cause some of the formulas to give inaccurate results if the table space of the table was converted to large but the indexes have not yet been recreated or reorganized.

**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, followed by the REORGCHK command.

**Note:** This formula is not supported for indexes on XML columns.

**F6** Results of Formula 6. The notation +++ indicates that the result exceeds 999, and might be invalid. Rerun REORGCHK with the UPDATE

## REORGCHK

STATISTICS option, or issue RUNSTATS, followed by the REORGCHK command. If the statistics are current and valid, you should reorganize.

**Note:** This formula is not supported for indexes on XML columns.

F7 Results of Formula 7.

F8 Results of Formula 8.

### 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 column corresponds to F4 (Formula 4)
- - or \* in the second from left column corresponds to F5 (Formula 5)
- - or \* in the middle column corresponds to F6 (Formula 6).
- - or \* in the second column from the right corresponds to F7 (Formula 7)
- - or \* on the right column corresponds to F8 (Formula 8).

Index reorganization advice is as follows:

- If the results of the calculations for Formula 1,2 and 3 do not exceed the bounds set by the formula and the results of the calculations for Formula 4,5 or 6 do exceed the bounds set, then index reorganization is recommended.
- If only the results of the calculations Formula 7 exceed the bounds set, but the results of Formula 1,2,3,4,5 and 6 are within the set bounds, then cleanup of the indexes using the CLEANUP ONLY option of index reorganization is recommended.
- If the only calculation result to exceed the set bounds is the that of Formula 8, then a cleanup of the pseudo empty pages of the indexes using the CLEANUP ONLY PAGES option of index reorganization is recommended.

On a partitioned table the results for formulas (5 to 8) can be misleading if when the statistics are collected, there are index keys in the non-partitioned index belonging to detached data partitions which require cleanup. When there are detached partitions on a partitioned table, such index keys will not be counted as part of the keys in the statistics because they are invisible and no longer part of the table. They will eventually get removed from the index by asynchronous index cleanup. As a result, statistics collected before asynchronous index cleanup is run will be misleading. If the **REORGCHK** command is issued before asynchronous index cleanup completes, it will likely generate a false alarm for index reorganization or index cleanup based on the inaccurate statistics. Once asynchronous index cleanup is run, all the index keys that still belong to detached data partitions which require cleanup will be removed and this may eliminate the need for index reorganization.

For partitioned tables, you are encouraged to issue the **REORGCHK** after an asynchronous index cleanup has completed in order to generate accurate index statistics in the presence of detached data partitions. To determine whether or not there are detached data partitions in the table, you can check the status field in the SYSDATAPARTITIONS table and look for the value I (index cleanup) or D (detached with dependant MQT).

**Usage notes:**

This command does not display declared temporary table statistical information.

This utility does not support the use of nicknames.

Unless you specify the CURRENT STATISTICS option, REORGCHK gathers statistics on all columns using the default options only. Specifically, column group are not gathered and if LIKE statistics were previously gathered, they are not gathered by REORGCHK. The statistics gathered depend on the kind of statistics currently stored in the catalog tables:

- If detailed index statistics are present in the catalog for any index, table statistics and detailed index statistics (without sampling) for all indexes are collected.
- If detailed index statistics are not detected, table statistics as well as regular index statistics are collected for every index.
- If distribution statistics are detected, distribution statistics are gathered on the table. If distribution statistics are gathered, the number of frequent values and quantiles are based on the database configuration parameter settings.

REORGCHK calculates statistics obtained from eight different formulas to determine if performance has deteriorated or can be improved by reorganizing a table or its indexes. When a table uses less than or equal to ( NPARTITIONS \* 1 extent size ) of pages, no table reorganization is recommended based on each formula. More specifically,

- For non-partitioned tables ( NPARTITIONS =1 ), the threshold is:  
(FPAGES <= 1 extent size)
- For partitioned tables, it is:  
(FPAGES <= NPARTITIONS \* 1 extent size)
- In a multi-partitioned database, after the number of database partitions in a database partition group of the table is factored in, this threshold for not recommending table reorganization changes to:  
FPAGES <= 'number of database partitions in a database partition group of the table' \* NPARTITIONS \* 1 extent size

Long field or LOB data is not accounted for while calculating TSIZE.

REORGCHK uses the following formulas to analyze the physical location of rows and the size of the table:

- Formula F1:

$$100 * \text{OVERFLOW} / \text{CARD} < 5$$

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:

For regular tables:

$$100 * \text{TSIZE} / ((\text{FPAGES} - \text{NPARTITIONS}) * (\text{TABLEPAGE} - 68)) > 70$$

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 68 bytes). Because the last page allocated in the data object is not usually filled, 1 is subtracted from FPAGES for each partition (which is the same as FPAGES-NPARTITIONS).

## REORGCHK

For MDC tables:

$$100 * TSIZE / ((ACTBLK - FULLKEYCARD) * EXTENTSIZE * (TABLEPAGESIZE - 68)) > 70$$

FULLKEYCARD represents the cardinality of the composite dimension index for the MDC table. Extent size is the number of pages per block. The formula checks if the table size in bytes is more than the 70 percent of the remaining blocks for a table after subtracting the minimum required number of blocks.

- 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.) As noted above, no table reorganization is recommended when  $(FPAGES \leq NPARTITIONS * 1 \text{ extent size})$ . Therefore, F3 is not calculated. For non-partitioned tables,  $NPARTITIONS = 1$ . In a multi-partitioned database, this condition changes to  $FPAGES = \text{'number of database partitions in a database partition group of the table'} * NPARTITIONS * 1 \text{ extent size}$ .

For MDC tables, the formula is:

$$100 * \text{activeblocks} / ((\text{fpages} / \text{ExtentSize}) - 1)$$

REORGCHK uses the following formulas to analyze the indexes and their the relationship to the table data:

- Formula F4:

– For non-partitioned tables:

$$\text{CLUSTERRATIO or normalized CLUSTERFACTOR} > 80$$

The global CLUSTERFACTOR and CLUSTERRATIO take into account the correlation between the index key and distribution key. 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.

– For partitioned tables:

$$\text{AVGPARTITION\_CLUSTERRATIO or normalized AVGPARTITION\_CLUSTERFACTOR} > 80$$

AVGPARTITION\_CLUSTERFACTOR and AVGPARTITION\_CLUSTERFACTOR values reflect how clustered data is within a data partition with respect to an index key. A partitioned table can be perfectly clustered for a particular index key within each data partition, and still have a low value for the CLUSTERFACTOR and CLUSTERRATIO because the index key is not a prefix of the table partitioning key. Design your tables and indexes using the most important index keys as a prefix of the table partitioning key. In addition, because the optimizer uses the global clusteredness values to make decisions about queries that span multiple data partitions, it is possible to perform a clustering reorganization and have the optimizer still not choose the clustering index when the keys do not agree.

- Formula F5:

$$100 * ( \text{KEYS} * (\text{LEAF\_RECSIZE} + \text{LEAF\_RECSIZE\_OVERHEAD}) + (\text{INDCARD} - \text{KEYS}) * \text{DUPKEYSIZE} ) / ( (\text{NLEAF} - \text{NUM\_EMPTY\_LEAFS} - 1) * (\text{INDEXPAGESIZE} - \text{LEAF\_PAGE\_OVERHEAD}) ) > \text{MIN}(50, (100 - \text{PCTFREE}))$$

The space in use at the leaf level of the index should be greater than the minimum of 50 and  $100 - \text{PCTFREE}$  percent (only checked when  $\text{NLEAF} > 1$ ).

- Formula F6:

$$\begin{aligned} & (100 - \text{PCTFREE}) * ( (\text{FLOOR}((100 - \text{LEVEL2PCTFREE}) / 100 * \\ & (\text{INDEXPAGESIZE} - \text{NLEAF\_PAGE\_OVERHEAD}) / (\text{NLEAF\_RECSIZE} + \text{NLEAF\_RECSIZE\_OVERHEAD}))) * \\ & (\text{FLOOR}((100 - \text{MIN}(10, \text{LEVEL2PCTFREE})) / 100 * (\text{INDEXPAGESIZE} - \text{NLEAF\_PAGE\_OVERHEAD}) / \\ & (\text{NLEAF\_RECSIZE} + \text{NLEAF\_RECSIZE\_OVERHEAD})) ** (\text{NLEVELS} - 3)) * \\ & (\text{INDEXPAGESIZE} - \text{LEAF\_PAGE\_OVERHEAD}) / (\text{KEYS} * (\text{LEAF\_RECSIZE} + \text{LEAF\_RECSIZE\_OVERHEAD}) + \\ & (\text{INDCARD} - \text{KEYS}) * \text{DUPKEYSIZE} ) ) < 100 \end{aligned}$$

To determine if recreating the index would result in a tree having fewer levels. This formula checks the ratio between the amount of space in an index tree that has one less level than the current tree, and the amount of space needed. If a tree with one less level could be created and still leave PCTFREE available, then a reorganization is recommended. The actual number of index entries should be more than (100-PCTFREE) percent of the number of entries an NLEVELS-1 index tree can handle (only checked if NLEVELS>2). In the case where NLEVELS = 2, the other REORGCHK formulas should be relied upon to determine if the index should be reorganized.

In simplified form, formula F6 can be rewritten as the following:

$$\frac{\text{Amount of space needed for an index if it was one level smaller}}{\text{Amount of space needed for all the entries in the index}} < 1$$

When the above left part is > 1, it means all index entries in the existing index can fit into an index that is one level smaller than the existing index. In this case, a reorg index is recommended.

The amount of space needed for an NLEVELS-1 index is calculated by:

(The max number of leaf pages that a NLEVELS-1 index can have) \*  
(Amount of space available to store index entries per leaf page)

where,

The max number of leaf pages that a NLEVELS-1 index can have =  
(No. of entries a level 2 index page can have) \*  
(No. of entries per page on levels greater than 2) \*\*  
(No. of levels in the intended index - 2) =

$$\begin{aligned} & \{ \text{FLOOR} \left( \left[ \frac{(100 - \text{LEVEL2PCTFREE})}{100} \right] * \right. \\ & \left. \left[ \frac{(\text{PageSize} - \text{Overhead})}{(\text{Avg. size of each nonleaf index entry})} \right] \right) * \\ & \text{FLOOR} \left( \left[ \frac{(100 - \text{MIN}(10, \text{LEVEL2PCTFREE}))}{100} \right] * \right. \\ & \left. \left[ \frac{(\text{PageSize} - \text{Overhead})}{(\text{Avg. size of each nonleaf index entry})} \right] \right) ** \\ & (\text{NLEVELS} - 3) \} \end{aligned}$$

(100 - LEVEL2PCTFREE) is the percentage of used space on level 2 of the index.

Level 2 is the level immediately above the leaf level.

(100 - MIN(10, LEVEL2PCTFREE)) is the percentage of used space on all levels above the second level.

NLEVELS is the number of index levels in the existing index.

The amount of space available to store index entries per leaf page =  
((100-PCTFREE)/100 \* (INDEXPAGESIZE - LEAF\_PAGE\_OVERHEAD)) =

## REORGCHK

( Used space per page \* (PageSize - Overhead) )

The amount of space needed for all index entries:

$KEYS * (LEAF\_RECSIZE + LEAF\_RECSIZE\_OVERHEAD) +$   
 $(INDCARD - KEYS) * DUPKEYSIZE$

$(KEYS * (LEAF\_RECSIZE + LEAF\_RECSIZE\_OVERHEAD))$  represents the space used for the first occurrence of each key value in the index and  $((INDCARD - KEYS) * DUPKEYSIZE)$  represents the space used for subsequent (duplicate) occurrences of a key value.

- Formula F7:

$100 * (NUMRIDS\_DELETED / (NUMRIDS\_DELETED + INDCARD)) < 20$

The number of pseudo-deleted RIDs on non-pseudo-empty pages should be less than 20 percent.

- Formula F8:

$100 * (NUM\_EMPTY\_LEAFS/NLEAF) < 20$

The number of pseudo-empty leaf pages should be less than 20 percent of the total number of leaf pages.

Running statistics on many tables can take time, especially if the tables are large.

### Related concepts:

- “Table reorganization” in *Performance Guide*

### Related reference:

- “REORGCHK\_IX\_STATS procedure – Retrieve index statistics for reorganization evaluation” in *Administrative SQL Routines and Views*
- “REORGCHK\_TB\_STATS procedure – Retrieve table statistics for reorganization evaluation” in *Administrative SQL Routines and Views*



## RESET ADMIN CONFIGURATION

Resets entries in the DB2 Administration Server (DAS) configuration file on the node to which you are connected. The DAS is a special administrative tool that enables remote administration of DB2 servers. The values are reset by node type, which is always a server with remote clients. For a list of DAS parameters, see the description of the UPDATE ADMINISTRATION CONFIGURATION command.

**Scope:**

This command resets the DAS configuration file on the administration node of the system to which you are connected.

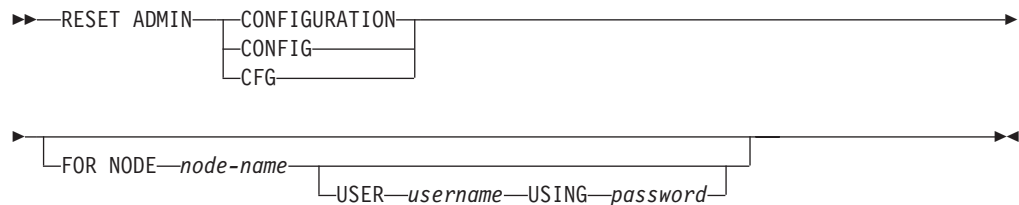
**Authorization:**

*dasadm*

**Required connection:**

Partition. To reset the DAS configuration for a remote system, specify the system using the FOR NODE option with the administration node name.

**Command syntax:**



**Command parameters:**

**FOR NODE**

Enter the name of an administrator node to reset DAS configuration parameters there.

**USER username USING password**

If connection to the remote system requires user name and password, enter this information.

**Usage notes:**

To reset the DAS configuration parameters on a remote system, specify the system using the administrator node name as an argument to the FOR NODE option and specify the user name and password if the connection to that node requires username and password authorization.

To view or print a list of the DAS configuration parameters, use the GET ADMIN CONFIGURATION command. To change the value of an admin parameter, use the UPDATE ADMIN CONFIGURATION command.

Changes to the DAS configuration parameters that can be updated on-line take place immediately. Other changes become effective only after they are loaded into memory when you restart the DAS with the **db2admin** command.

## RESET ADMIN CONFIGURATION

If an error occurs, the DAS configuration file does not change.

The DAS configuration file cannot be reset if the checksum is invalid. This might occur if you edit the DAS configuration file manually and do not use the appropriate command. If the checksum is invalid, you must drop and recreate the DAS to reset the its configuration file.

**Related reference:**

- “GET ADMIN CONFIGURATION ” on page 441
- “UPDATE ADMIN CONFIGURATION ” on page 756
- “Configuration parameters summary” in *Performance Guide*

## RESET ALERT CONFIGURATION

Resets the health indicator settings for specific objects to the current defaults for that object type or resets the current default health indicator settings for an object type to the install defaults.

### Authorization:

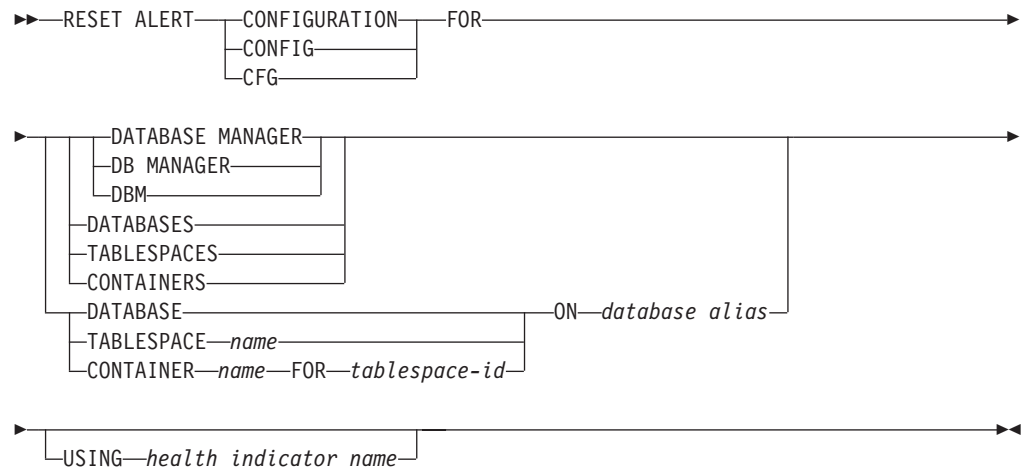
One of the following:

- sysadm
- sysmaint
- sysctrl

### Required connection:

Instance. An explicit attachment is not required.

### Command syntax:



### Command parameters:

#### DATABASE MANAGER

Resets alert settings for the database manager.

#### DATABASES

Resets alert settings for all databases managed by the database manager. These are the settings that apply to all databases that do not have custom settings. Custom settings are defined using the `DATABASE ON database alias` clause.

#### CONTAINERS

Resets default alert settings for all table space containers managed by the database manager to the install default. These are the settings that apply to all table space containers that do not have custom settings. Custom settings are defined using the `"CONTAINER name ON database alias"` clause.

#### CONTAINER name FOR tablespace-id FOR tablespace-id ON database alias

Resets the alert settings for the table space container called `name`, for the table space specified using the `"FOR tablespace-id"` clause, on the database

## RESET ALERT CONFIGURATION

specified using the "ON *database alias*" clause. If this table space container has custom settings, then these settings are removed and the current table space containers default is used.

### TABLESPACES

Resets default alert settings for all table spaces managed by the database manager to the install default. These are the settings that apply to all table spaces that do not have custom settings. Custom settings are defined using the "TABLESPACE *name* ON *database alias*" clause.

### DATABASE ON *database alias*

Resets the alert settings for the database specified using the ON *database alias* clause. If this database has custom settings, then these settings are removed and the install default is used.

### TABLESPACE *name* ON *database alias*

Resets the alert settings for the table space called *name*, on the database specified using the ON *database alias* clause. If this table space has custom settings, then these settings are removed and the install default is used.

### USING *health indicator name*

Specifies the set of health indicators for which alert configuration will be reset. Health indicator names consist of a two-letter object identifier followed by a name that describes what the indicator measures. For example:

```
db.sort_privmem_util
```

If you do not specify this option, all health indicators for the specified object or object type will be reset.

### Related tasks:

- "Configuring health indicators using a client application" in *System Monitor Guide and Reference*

### Related reference:

- "db2ResetAlertCfg API - Reset the alert configuration of health indicators" in *Administrative API Reference*
- "RESET ALERT CONFIGURATION command using the ADMIN\_CMD procedure" in *Administrative SQL Routines and Views*
- "HEALTH\_GET\_ALERT\_ACTION\_CFG table function –Retrieve health alert action configuration settings" in *Administrative SQL Routines and Views*
- "HEALTH\_GET\_ALERT\_CFG table function – Retrieve health alert configuration settings" in *Administrative SQL Routines and Views*
- "UPDATE ALERT CONFIGURATION " on page 758

## 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:**

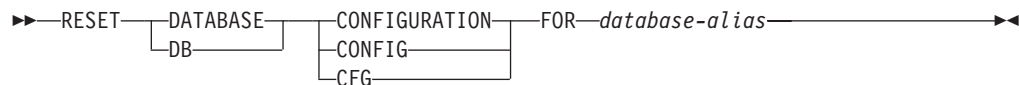
One of the following:

- *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:**



**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 the GET DATABASE CONFIGURATION command.

To change the value of a configurable parameter, use the UPDATE DATABASE CONFIGURATION command.

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 might 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.

The **RESET DATABASE CONFIGURATION** command will reset the database configuration parameters to the pre-database configuration values, where

## RESET DATABASE CONFIGURATION

AUTO\_RUNSTATS will be ON. SELF\_TUNING\_MEMORY will be reset to ON on non-partitioned database environments and to OFF on partitioned database environments.

### Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

### Related reference:

- “GET DATABASE CONFIGURATION ” on page 457
- “UPDATE DATABASE CONFIGURATION ” on page 772
- “db2CfgSet API - Set the database manager or database configuration parameters” in *Administrative API Reference*
- “Configuration parameters summary” in *Performance Guide*
- “RESET DATABASE CONFIGURATION command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “DBCFCG administrative view – Retrieve database configuration parameter information” in *Administrative SQL Routines and Views*

## RESET DATABASE MANAGER CONFIGURATION

Resets the parameters in the database manager configuration file to the system defaults. The values are reset by node type.

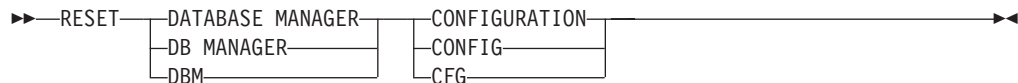
**Authorization:**

*sysadm*

**Required connection:**

None or instance. An instance attachment is not required to perform local database manager configuration operations, but is required to perform remote database manager configuration operations. To update the database manager configuration for a remote instance, it is necessary to first attach to that instance. To update a configuration parameter on-line, it is also necessary to first attach to the instance.

**Command syntax:**



**Command parameters:**

None

**Usage notes:**

This command resets all parameters set by the installation program. This could cause error messages to be returned when restarting DB2. For example, if the SVCENAME parameter is reset, the user will receive the SQL5043N error message when trying to restart DB2.

Before running this command, save the output from the GET DATABASE MANAGER CONFIGURATION command to a file so that you can refer to the existing settings. Individual settings can then be updated using the UPDATE DATABASE MANAGER CONFIGURATION command.

It is not recommended that the SVCENAME parameter, set by the installation program, be modified by the user.

To view or print a list of the database manager configuration parameters, use the GET DATABASE MANAGER CONFIGURATION command. To change the value of a configurable parameter, use the UPDATE DATABASE MANAGER CONFIGURATION command.

For more information about these parameters, refer to the summary list of configuration parameters and the individual parameters.

Some changes to the database manager configuration file become effective only after they are loaded into memory. For more information on which parameters are configurable on-line and which ones are not, see the configuration parameter summary. Server configuration parameters that are not reset immediately are reset during execution of **db2start**. For a client configuration parameter, parameters are reset the next time you restart the application. If the client is the command line processor, it is necessary to invoke **TERMINATE**.

## RESET DATABASE MANAGER CONFIGURATION

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 might occur if the database manager you edit the configuration file manually and do not use the appropriate command. If the checksum is invalid, you must reinstall the database manager to reset the database manager configuration file

### Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

### Related reference:

- “GET DATABASE MANAGER CONFIGURATION ” on page 463
- “TERMINATE ” on page 744
- “UPDATE DATABASE MANAGER CONFIGURATION ” on page 775
- “db2CfgSet API - Set the database manager or database configuration parameters” in *Administrative API Reference*
- “Configuration parameters summary” in *Performance Guide*
- “RESET DATABASE MANAGER CONFIGURATION command using the ADMIN\_CMD procedure” in *Administrative SQL Routines and Views*
- “DBMCFG administrative view – Retrieve database manager configuration parameter information” in *Administrative SQL Routines and Views*



## 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:

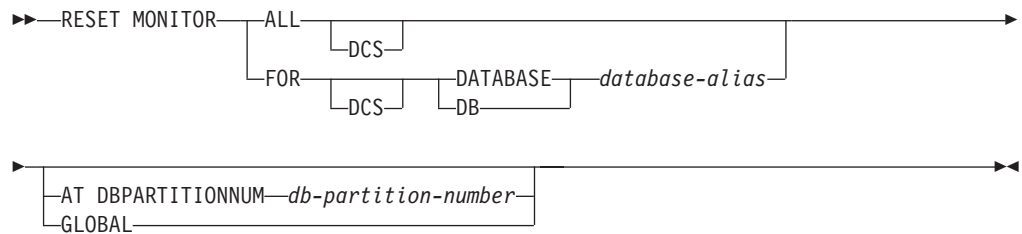
- *sysadm*
- *sysctrl*
- *sysmaint*
- *sysmon*

### Required connection:

Instance. If there is no instance attachment, a default instance attachment is created.

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:



### 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.

#### AT DBPARTITIONNUM db-partition-number

Specifies the database partition for which the status of the monitor switches is to be displayed.

**global** Returns an aggregate result for all database partitions in a partitioned database environment.

### Usage notes:

## RESET MONITOR

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.

If ALL is specified, some database manager information is also reset to maintain consistency of the returned data, and some database partition-level counters are reset.

### **Compatibilities:**

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.

### **Related reference:**

- "db2ResetMonitor API - Reset the database system monitor data" in *Administrative API Reference*
- "GET MONITOR SWITCHES " on page 478
- "GET SNAPSHOT " on page 487

## 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.

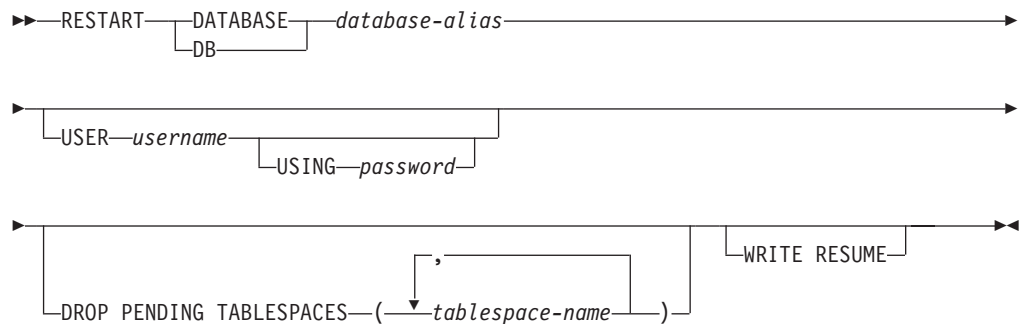
### Authorization:

None

### Required connection:

This command establishes a database connection.

### Command syntax:



### 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. If a table space is in the drop-pending state, the only possible action is to drop the table space.

In the case of circular logging, a troubled table space will cause a restart failure. A list of troubled table space names can be found in the administration notification log if a restart database operation fails because of container problems. If there is only one system temporary table space in

## RESTART DATABASE

the database, and it is in drop pending state, a new system temporary table space must be created immediately following a successful database restart operation.

### WRITE RESUME

Allows you to force a database restart on databases that failed while I/O writes were suspended. Before performing crash recovery, this option will resume I/O writes by removing the SUSPEND\_WRITE state from every table space in the database.

The WRITE RESUME option can also be used in the case where the connection used to suspend I/O writes is currently hung and all subsequent connection attempts are also hanging. When used in this circumstance, RESTART DATABASE will resume I/O writes to the database without performing crash recovery. RESTART DATABASE with the WRITE RESUME option will only perform crash recovery when you use it after a database crash. The WRITE RESUME parameter can only be applied to the primary database, not to mirrored databases.

### 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 the LIST INDOUBT TRANSACTIONS command to generate a list of indoubt transactions.

If the database is only restarted on a single node within an MPP system, a message might 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.

### Related tasks:

- “Resolving indoubt transactions manually” in *Administration Guide: Planning*

### Related reference:

- “LIST INDOUBT TRANSACTIONS ” on page 538
- “db2DatabaseRestart API - Restart database” in *Administrative API Reference*

---

## RESTORE DATABASE

The **RESTORE DATABASE** command recreates a damaged or corrupted database that has been backed up using the DB2 backup utility. The restored database is in the same state that it was in when the backup copy was made. This utility can also overwrite a database with a different image or restore the backup copy to a new database.

For information on the restore operations supported by DB2 database systems between different operating systems and hardware platforms, see "Backup and restore operations between different operating systems and hardware platforms" in the *Related concepts* section.

The restore utility can also be used to restore backup images that were produced on DB2 UDB Version 8. If a migration is required, it will be invoked automatically at the end of the restore operation.

If, at the time of the backup operation, the database was enabled for rollforward recovery, the database can be brought to its previous state by invoking the rollforward utility after successful completion of a restore operation.

This utility can also restore a table space level backup.

Incremental images and images only capturing differences from the time of the previous capture (called a "delta image") cannot be restored when there is a difference in operating systems or word size (32-bit or 64-bit).

Following a successful restore operation from one environment to a different environment, no incremental or delta backups are allowed until a non-incremental backup is taken. (This is not a limitation following a restore operation within the same environment.)

Even with a successful restore operation from one environment to a different environment, there are some considerations: packages must be rebound before use (using the BIND command, the REBIND command, or the db2rbind utility); SQL procedures must be dropped and re-created; and all external libraries must be rebuilt on the new platform. (These are not considerations when restoring to the same environment.)

### 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*

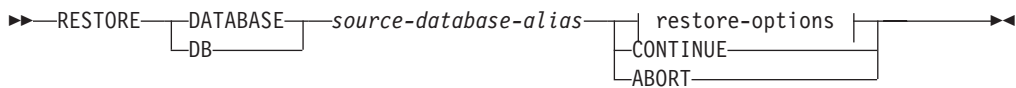
# RESTORE DATABASE

## Required connection:

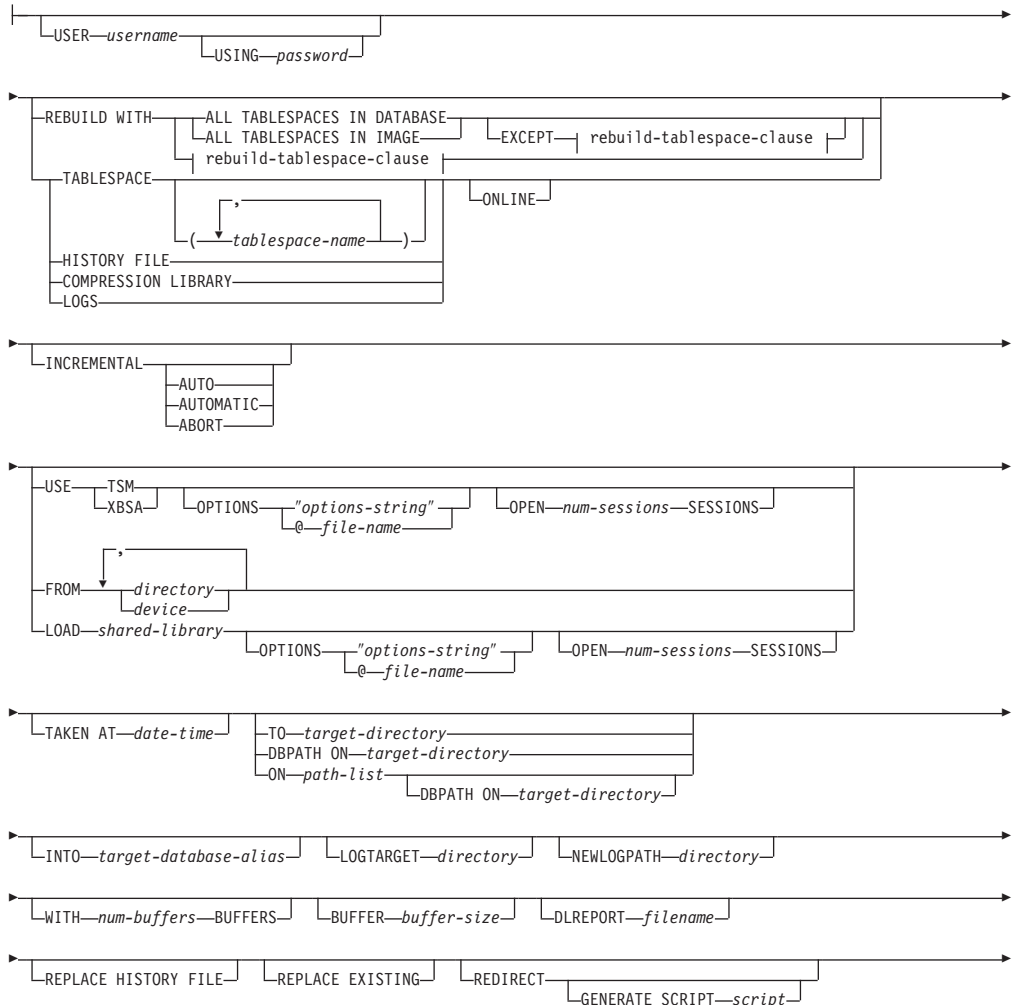
The required connection will vary based on the type of restore action:

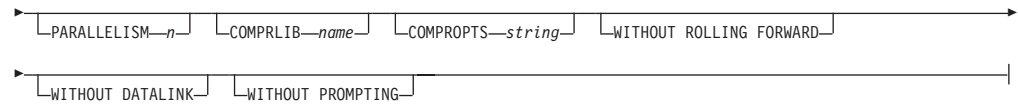
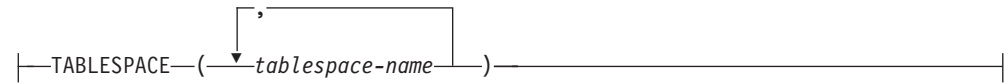
- You require a database connection, to restore to an existing database. This command automatically establishes an exclusive connection to the specified database.
- You require an instance and a database connection, to restore to a new database. The instance attachment is required to create the database.  
To restore to a new database at an instance different from the current instance, it is necessary to first attach to the instance where the new database will reside. The new instance can be local or remote. The current instance is defined by the value of the DB2INSTANCE environment variable.

## Command syntax:



## restore-options:



**rebuild-tablespace-clause:****Command parameters:****DATABASE** *source-database-alias*

Alias of the source database from which the backup was taken.

**CONTINUE**

Specifies that the containers have been redefined, and that the final step in a redirected restore operation should be performed.

**ABORT**

This parameter:

- Stops a redirected restore operation. This is useful when an error has occurred that requires one or more steps to be repeated. After RESTORE DATABASE with the ABORT option has been issued, each step of a redirected restore operation must be repeated, including RESTORE DATABASE with the REDIRECT option.
- Terminates an incremental restore operation before completion.

**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.

**REBUILD WITH ALL TABLESPACES IN DATABASE**

Restores the database with all the table spaces known to the database at the time of the image being restored. This restore overwrites a database if it already exists.

**REBUILD WITH ALL TABLESPACES IN DATABASE EXCEPT***rebuild-tablespace-clause*

Restores the database with all the table spaces known to the database at the time of the image being restored except for those specified in the list. This restore overwrites a database if it already exists.

**REBUILD WITH ALL TABLESPACES IN IMAGE**

Restores the database with only the table spaces in the image being restored. This restore overwrites a database if it already exists.

**REBUILD WITH ALL TABLESPACES IN IMAGE EXCEPT** *rebuild-tablespace-clause*

Restores the database with only the table spaces in the image being restored except for those specified in the list. This restore overwrites a database if it already exists.

**REBUILD WITH** *rebuild-tablespace-clause*

Restores the database with only the list of table spaces specified. This restore overwrites a database if it already exists.

## RESTORE DATABASE

### **TABLESPACE** *tablespace-name*

A list of names used to specify the table spaces that are to be restored.

### **ONLINE**

This keyword, applicable only when performing a table space-level restore operation, is specified to allow a backup image to be restored online. This means that other agents can connect to the database while the backup image is being restored, and that the data in other table spaces will be available while the specified table spaces are being restored.

### **HISTORY FILE**

This keyword is specified to restore only the history file from the backup image.

### **COMPRESSION LIBRARY**

This keyword is specified to restore only the compression library from the backup image. If the object exists in the backup image, it will be restored into the database directory. If the object does not exist in the backup image, the restore operation will fail.

**LOGS** This keyword is specified to restore only the set of log files contained in the backup image. If the backup image does not contain any log files, the restore operation will fail. If this option is specified, the LOGTARGET option must also be specified.

### **INCREMENTAL**

Without additional parameters, INCREMENTAL specifies a manual cumulative restore operation. During manual restore the user must issue each restore command manually for each image involved in the restore. Do so according to the following order: last, first, second, third and so on up to and including the last image.

### **INCREMENTAL AUTOMATIC/AUTO**

Specifies an automatic cumulative restore operation.

### **INCREMENTAL ABORT**

Specifies abortion of an in-progress manual cumulative restore operation.

### **USE TSM**

Specifies that the database is to be restored from TSM-managed output.

### **OPTIONS**

*"options-string"*

Specifies options to be used for the restore operation. The string will be passed to the vendor support library, for example TSM, exactly as it was entered, without the quotes. Specifying this option overrides the value specified by the VENDOROPT database configuration parameter.

*@file-name*

Specifies that the options to be used for the restore operation are contained in a file located on the DB2 server. The string will be passed to the vendor support library, for example TSM. The file must be a fully qualified file name.

### **OPEN** *num-sessions* **SESSIONS**

Specifies the number of I/O sessions that are to be used with TSM or the vendor product.

### **USE XBSA**

Specifies that the XBSA interface is to be used. Backup Services APIs



(XBSA) are an open application programming interface for applications or facilities needing data storage management for backup or archiving purposes.

**FROM** *directory/device*

The fully qualified path name of the directory or device on which the backup image resides. If USE TSM, FROM, and LOAD are omitted, the default value is the current working directory of the client machine. This target directory or device must exist on the target server/instance.

On Windows operating systems, 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
```

Using these commands, the DB2 database system generates subdirectories under the c:\backup directory to allow more than one backup to be placed in the specified top level directory. The DB2 generated subdirectories should be ignored. To specify precisely which backup image to restore, use the TAKEN AT parameter. There can be several backup images stored on 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, continue when a new tape has been mounted).
- d** Device terminate. Stop using *only* the device that generated the warning message (for example, terminate when there are no more tapes).
- t** Terminate. Abort the restore operation after the user has failed to perform some action requested by the utility.

**LOAD** *shared-library*

The name of the shared library (DLL on Windows operating systems) containing the vendor backup and restore I/O functions to be used. The name can contain a full path. If the full path is not given, the value defaults to the path on which the user exit program resides.

**TAKEN AT** *date-time*

The time stamp of the database backup image. The time stamp is displayed after successful completion of a backup operation, and is part of the path name for the backup image. It is specified in the form *yyyymmddhhmmss*. A partial time stamp can also be specified. For example, if two different backup images with time stamps 20021001010101 and 20021002010101 exist, specifying 20021002 causes the image with time stamp 20021002010101 to be used. If a value for this parameter is not specified, there must be only one backup image on the source media.

**TO** *target-directory*

This parameter states the target database directory. This parameter is ignored if the utility is restoring to an existing database. The drive and directory that you specify must be local. If the backup image contains a database that is enabled for automatic storage then only the database directory changes, the storage paths associated with the database do not change.

## RESTORE DATABASE

### DBPATH ON *target-directory*

This parameter states the target database directory. This parameter is ignored if the utility is restoring to an existing database. The drive and directory that you specify must be local. If the backup image contains a database that is enabled for automatic storage and the ON parameter is not specified then this parameter is synonymous with the TO parameter and only the database directory changes, the storage paths associated with the database do not change.

### ON *path-list*

This parameter redefines the storage paths associated with an automatic storage database. Using this parameter with a database that is not enabled for automatic storage results in an error (SQL20321N). The existing storage paths as defined within the backup image are no longer used and automatic storage table spaces are automatically redirected to the new paths. If this parameter is not specified for an automatic storage database then the storage paths remain as they are defined within the backup image.

One or more paths can be specified, each separated by a comma. Each path must have an absolute path name and it must exist locally. If the database does not already exist on disk and the DBPATH ON parameter is not specified then the first path is used as the target database directory.

For a multi-partition database the ON path-list option can only be specified on the catalog partition. The catalog partition must be restored before any other partitions are restored when the ON option is used. The restore of the catalog-partition with new storage paths will place all non-catalog nodes in a RESTORE\_PENDING state. The non-catalog nodes can then be restored in parallel without specifying the ON clause in the restore command.

In general, the same storage paths must be used for each partition in a multi-partition database and they must all exist prior to executing the **RESTORE DATABASE** command. One exception to this is where database partition expressions are used within the storage path. Doing this allows the database partition number to be reflected in the storage path such that the resulting path name is different on each partition.

You use the argument " \$N" ([blank]\$N) to indicate a database partition expression. A database partition expression can be used anywhere in the storage path, and multiple database partition expressions can be specified. Terminate the database partition expression with a space character; whatever follows the space is appended to the storage path after the database partition expression is evaluated. If there is no space character in the storage path after the database partition expression, it is assumed that the rest of the string is part of the expression. The argument can only be used in one of the following forms:

*Table 14.* . Operators are evaluated from left to right. % represents the modulus operator. The database partition number in the examples is assumed to be 10.

Syntax	Example	Value
[blank]\$N	" \$N"	10
[blank]\$N+[number]	" \$N+100"	110
[blank]\$N%[number]	" \$N%5"	0
[blank]\$N+[number]%[number]	" \$N+1%5"	1
[blank]\$N%[number]+[number]	" \$N%4+2"	4

Table 14. (continued). Operators are evaluated from left to right. % represents the modulus operator. The database partition number in the examples is assumed to be 10.

Syntax	Example	Value
<sup>a</sup> % is modulus.		

**INTO** *target-database-alias*

The target database alias. If the target database does not exist, it is created.

When you restore a database backup to an existing database, the restored database inherits the alias and database name of the existing database.

When you restore a database backup to a nonexistent database, the new database is created with the alias and database name that you specify. This new database name must be unique on the system where you restore it.

**LOGTARGET** *directory*

The absolute path name of an existing directory on the database server, to be used as the target directory for extracting log files from a backup image. If this option is specified, any log files contained within the backup image will be extracted into the target directory. If this option is not specified, log files contained within a backup image will not be extracted. To extract only the log files from the backup image, specify the LOGS option.

**NEWLOGPATH** *directory*

The absolute pathname of a directory that will be used for active log files after the restore operation. This parameter has the same function as the *newlogpath* database configuration parameter, except that its effect is limited to the restore operation in which it is specified. The parameter can be used when the log path in the backup image is not suitable for use after the restore operation; for example, when the path is no longer valid, or is being used by a different database.

**WITH** *num-buffers* **BUFFERS**

The number of buffers to be used. The DB2 database system will automatically choose an optimal value for this parameter unless you explicitly enter a value. A larger number of buffers can be used to improve performance when multiple sources are being read from, or if the value of PARALLELISM has been increased.

**BUFFER** *buffer-size*

The size, in pages, of the buffer used for the restore operation. The DB2 database system will automatically choose an optimal value for this parameter unless you explicitly enter a value. The minimum value for this parameter is 8 pages.

The restore buffer size must be a positive integer multiple of the backup buffer size specified during the backup operation. If an incorrect buffer size is specified, the buffers are allocated to be of the smallest acceptable size.

**DLREPORT** *filename*

The file name, if specified, must be specified as an absolute path. Reports the files that become unlinked, as a result of a fast reconcile, during a restore operation. This option is only to be used if the table being restored has a DATALINK column type and linked files.

**REPLACE HISTORY FILE**

Specifies that the restore operation should replace the history file on disk with the history file from the backup image.

**REPLACE EXISTING**

If a database with the same alias as the target database alias already exists,

## RESTORE DATABASE

this parameter specifies that the restore utility is to replace the existing database with the restored database. This is useful for scripts that invoke the restore utility, because the command line processor will not prompt the user to verify deletion of an existing database. If the WITHOUT PROMPTING parameter is specified, it is not necessary to specify REPLACE EXISTING, but in this case, the operation will fail if events occur that normally require user intervention.

### REDIRECT

Specifies a redirected restore operation. To complete a redirected restore operation, this command should be followed by one or more SET TABLESPACE CONTAINERS commands, and then by a RESTORE DATABASE command with the CONTINUE option. All commands associated with a single redirected restore operation must be invoked from the same window or CLP session. A redirected restore operation cannot be performed against a table space that has automatic storage enabled.

### GENERATE SCRIPT *script*

Creates a redirect restore script with the specified file name. The script name can be relative or absolute and the script will be generated on the client side. If the file cannot be created on the client side, an error message (SQL9304N) will be returned. If the file already exists, it will be overwritten. Please see the examples below for further usage information.

### WITHOUT ROLLING FORWARD

Specifies that the database is not to be put in rollforward pending state after it has been successfully restored.

If, following a successful restore operation, the database is in rollforward pending state, the ROLLFORWARD command must be invoked before the database can be used again.

If this option is specified when restoring from an online backup image, error SQL2537N will be returned.

If backup image is of a recoverable database then WITHOUT ROLLING FORWARD cannot be specified with REBUILD option.

### WITHOUT DATALINK

Specifies that any tables with DATALINK columns are to be put 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 that are to be created during the restore operation. The DB2 database system will automatically choose an optimal value for this parameter unless you explicitly enter a value.

### COMPRLIB *name*

Indicates the name of the library to be used to perform the decompression. The name must be a fully qualified path referring to a file on the server. If this parameter is not specified, DB2 will attempt to use the library stored in the image. If the backup was not compressed, the value of this parameter will be ignored. If the specified library cannot be loaded, the restore operation will fail.

### COMPROPTS *string*

Describes a block of binary data that is passed to the initialization routine in the decompression library. The DB2 database system passes this string directly from the client to the server, so any issues of byte reversal or code page conversion are handled by the decompression library. If the first

character of the data block is "@", the remainder of the data is interpreted by the DB2 database system as the name of a file residing on the server. The DB2 database system will then replace the contents of *string* with the contents of this file and pass the new value to the initialization routine instead. The maximum length for the string is 1 024 bytes.

#### WITHOUT PROMPTING

Specifies that the restore operation is to run unattended. Actions that normally require user intervention will return an error message. When using a removable media device, such as tape or diskette, the user is prompted when the device ends, even if this option is specified.

#### Examples:

1. In the following example, the database WSDB is defined on all 4 database partitions, numbered 0 through 3. The path /dev3/backup is accessible from all database partitions. The following offline backup images are available from /dev3/backup:

```
wsdb.0.db2inst1.NODE0000.CATN0000.20020331234149.001
wsdb.0.db2inst1.NODE0001.CATN0000.20020331234427.001
wsdb.0.db2inst1.NODE0002.CATN0000.20020331234828.001
wsdb.0.db2inst1.NODE0003.CATN0000.20020331235235.001
```

To restore the catalog partition first, then all other database partitions of the WSDB database from the /dev3/backup directory, issue the following commands from one of the database partitions:

```
db2_all '<<+0< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234149
 INTO wsdb REPLACE EXISTING'
db2_all '<<+1< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234427
 INTO wsdb REPLACE EXISTING'
db2_all '<<+2< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331234828
 INTO wsdb REPLACE EXISTING'
db2_all '<<+3< db2 RESTORE DATABASE wsdb FROM /dev3/backup
TAKEN AT 20020331235235
 INTO wsdb REPLACE EXISTING'
```

The db2\_all utility issues the restore command to each specified database partition. When performing a restore using db2\_all, you should always specify REPLACE EXISTING and/or WITHOUT PROMPTING. Otherwise, if there is prompting, the operation will look like it is hanging. This is because db2\_all does not support user prompting.

2. Following is a typical redirected restore scenario for a database whose alias is MYDB:

- a. Issue a RESTORE DATABASE command with the REDIRECT option.
 

```
restore db mydb replace existing redirect
```

After successful completion of step 1, and before completing step 3, the restore operation can be aborted by issuing:

```
restore db mydb abort
```

- b. Issue a SET TABLESPACE CONTAINERS command for each table space whose containers must be redefined. For example:

```
set tablespace containers for 5 using
(file 'f:\ts3con1' 20000, file 'f:\ts3con2' 20000)
```

## RESTORE DATABASE

To verify that the containers of the restored database are the ones specified in this step, issue the `LIST TABLESPACE CONTAINERS` command.

- c. After successful completion of steps 1 and 2, issue:

```
restore db mydb continue
```

This is the final step of the redirected restore operation.

- d. If step 3 fails, or if the restore operation has been aborted, the redirected restore can be restarted, beginning at step 1.

3. Following is a sample weekly incremental backup strategy for a recoverable database. It includes a weekly full database backup operation, a daily non-cumulative (delta) backup operation, and a mid-week cumulative (incremental) backup operation:

```
(Sun) backup db mydb use tsm
(Mon) backup db mydb online incremental delta use tsm
(Tue) backup db mydb online incremental delta use tsm
(Wed) backup db mydb online incremental use tsm
(Thu) backup db mydb online incremental delta use tsm
(Fri) backup db mydb online incremental delta use tsm
(Sat) backup db mydb online incremental use tsm
```

For an automatic database restore of the images created on Friday morning, issue:

```
restore db mydb incremental automatic taken at (Fri)
```

For a manual database restore of the images created on Friday morning, issue:

```
restore db mydb incremental taken at (Fri)
restore db mydb incremental taken at (Sun)
restore db mydb incremental taken at (Wed)
restore db mydb incremental taken at (Thu)
restore db mydb incremental taken at (Fri)
```

4. To produce a backup image, which includes logs, for transportation to a remote site:

```
backup db sample online to /dev3/backup include logs
```

To restore that backup image, supply a `LOGTARGET` path and specify this path during `ROLLFORWARD`:

```
restore db sample from /dev3/backup logtarget /dev3/logs
rollforward db sample to end of logs and stop overflow log path /dev3/logs
```

5. To retrieve only the log files from a backup image that includes logs:
- ```
restore db sample logs from /dev3/backup logtarget /dev3/logs
```
6. The `USE TSM OPTIONS` keywords can be used to specify the TSM information to use for the restore operation. On Windows platforms, omit the `-fromowner` option.

- Specifying a delimited string:

```
restore db sample use TSM options '"-fromnode=bar -fromowner=dmcinnis"'
```

- Specifying a fully qualified file:

```
restore db sample use TSM options @/u/dmcinnis/myoptions.txt
```

The file `myoptions.txt` contains the following information: `-fromnode=bar -fromowner=dmcinnis`

7. The following is a simple restore of a multi-partition automatic storage enabled database with new storage paths. The database was originally created with one storage path, `/myPath0`:

- On the catalog partition issue: `restore db mydb on /myPath1,/myPath2`

- On all non-catalog partitions issue: restore db mydb
8. A script output of the following command on a non-auto storage database:
 restore db sample from /home/jseifert/backups taken at 20050301100417 redirect
 generate script SAMPLE_NODE0000.clp

would look like this:

```
-- *****
-- ** automatically created redirect restore script
-- *****
UPDATE COMMAND OPTIONS USING S ON Z ON SAMPLE_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM 0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
-- *****
-- ** initialize redirected restore
-- *****
RESTORE DATABASE SAMPLE
-- USER '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050301100417
-- DBPATH ON '<target-directory>'
INTO SAMPLE
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00001/SQL0GDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- *****
-- ** tablespace definition
-- *****
-- *****
-- ** Tablespace name                = SYSCATSPACE
-- ** Tablespace ID                   = 0
-- ** Tablespace Type                  = System managed space
-- ** Tablespace Content Type         = Any data
-- ** Tablespace Page size (bytes)    = 4096
-- ** Tablespace Extent size (pages)  = 32
-- ** Using automatic storage         = No
-- ** Total number of pages           = 5572
-- *****
SET TABLESPACE CONTAINERS FOR 0
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH  'SQLT0000.0'
);
-- *****
-- ** Tablespace name                = TEMPSPACE1
-- ** Tablespace ID                   = 1
-- ** Tablespace Type                  = System managed space
-- ** Tablespace Content Type         = System Temporary data
-- ** Tablespace Page size (bytes)    = 4096
-- ** Tablespace Extent size (pages)  = 32
-- ** Using automatic storage         = No
-- ** Total number of pages           = 0
-- *****
SET TABLESPACE CONTAINERS FOR 1
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH  'SQLT0001.0'
);
```

RESTORE DATABASE

```
-- *****
-- ** Tablespace name                = USERSPACE1
-- ** Tablespace ID                  = 2
-- ** Tablespace Type                = System managed space
-- ** Tablespace Content Type        = Any data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage        = No
-- ** Total number of pages          = 1
-- *****
SET TABLESPACE CONTAINERS FOR 2
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  PATH 'SQLT0002.0'
);
-- *****
-- ** Tablespace name                = DMS
-- ** Tablespace ID                  = 3
-- ** Tablespace Type                = Database managed space
-- ** Tablespace Content Type        = Any data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage        = No
-- ** Auto-resize enabled             = No
-- ** Total number of pages          = 2000
-- ** Number of usable pages         = 1960
-- ** High water mark (pages)        = 96
-- *****
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  FILE /tmp/dms1 1000
, FILE /tmp/dms2 1000
);
-- *****
-- ** Tablespace name                = RAW
-- ** Tablespace ID                  = 4
-- ** Tablespace Type                = Database managed space
-- ** Tablespace Content Type        = Any data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage        = No
-- ** Auto-resize enabled             = No
-- ** Total number of pages          = 2000
-- ** Number of usable pages         = 1960
-- ** High water mark (pages)        = 96
-- *****
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
  DEVICE '/dev/hdb1' 1000
, DEVICE '/dev/hdb2' 1000
);
-- *****
-- ** start redirect restore
-- *****
RESTORE DATABASE SAMPLE CONTINUE;
-- *****
-- ** end of file
-- *****
```

9. A script output of the following command on an automatic storage database:
restore db test from /home/jseifert/backups taken at 20050304090733 redirect
generate script TEST_NODE0000.clp

would look like this:


```

-- *****
-- ** automatically created redirect restore script
-- *****
UPDATE COMMAND OPTIONS USING S ON Z ON TEST_NODE0000.out V ON;
SET CLIENT ATTACH_DBPARTITIONNUM 0;
SET CLIENT CONNECT_DBPARTITIONNUM 0;
-- *****
-- ** initialize redirected restore
-- *****
RESTORE DATABASE TEST
-- USER '<username>'
-- USING '<password>'
FROM '/home/jseifert/backups'
TAKEN AT 20050304090733
ON '/home/jseifert'
-- DBPATH ON <target-directory>
INTO TEST
-- NEWLOGPATH '/home/jseifert/jseifert/NODE0000/SQL00002/SQLLOGDIR/'
-- WITH <num-buff> BUFFERS
-- BUFFER <buffer-size>
-- REPLACE HISTORY FILE
-- REPLACE EXISTING
REDIRECT
-- PARALLELISM <n>
-- WITHOUT ROLLING FORWARD
-- WITHOUT PROMPTING
;
-- *****
-- ** tablespace definition
-- *****
-- *****
-- ** Tablespace name = SYSCATSPACE
-- ** Tablespace ID = 0
-- ** Tablespace Type = Database managed space
-- ** Tablespace Content Type = Any data
-- ** Tablespace Page size (bytes) = 4096
-- ** Tablespace Extent size (pages) = 4
-- ** Using automatic storage = Yes
-- ** Auto-resize enabled = Yes
-- ** Total number of pages = 6144
-- ** Number of usable pages = 6140
-- ** High water mark (pages) = 5968
-- *****
-- *****
-- ** Tablespace name = TEMPSPACE1
-- ** Tablespace ID = 1
-- ** Tablespace Type = System managed space
-- ** Tablespace Content Type = System Temporary data
-- ** Tablespace Page size (bytes) = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage = Yes
-- ** Total number of pages = 0
-- *****
-- *****
-- ** Tablespace name = USERSPACE1
-- ** Tablespace ID = 2
-- ** Tablespace Type = Database managed space
-- ** Tablespace Content Type = Any data
-- ** Tablespace Page size (bytes) = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage = Yes
-- ** Auto-resize enabled = Yes
-- ** Total number of pages = 256
-- ** Number of usable pages = 224
-- ** High water mark (pages) = 96
-- *****
-- *****

```

RESTORE DATABASE

```

-- ** Tablespace name                = DMS
-- ** Tablespace ID                  = 3
-- ** Tablespace Type                = Database managed space
-- ** Tablespace Content Type        = Any data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage         = No
-- ** Auto-resize enabled             = No
-- ** Total number of pages          = 2000
-- ** Number of usable pages         = 1960
-- ** High water mark (pages)        = 96
-- *****
SET TABLESPACE CONTAINERS FOR 3
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
    FILE '/tmp/dms1'                1000
    , FILE '/tmp/dms2'              1000
);
-- *****
-- ** Tablespace name                = RAW
-- ** Tablespace ID                  = 4
-- ** Tablespace Type                = Database managed space
-- ** Tablespace Content Type        = Any data
-- ** Tablespace Page size (bytes)   = 4096
-- ** Tablespace Extent size (pages) = 32
-- ** Using automatic storage         = No
-- ** Auto-resize enabled             = No
-- ** Total number of pages          = 2000
-- ** Number of usable pages         = 1960
-- ** High water mark (pages)        = 96
-- *****
SET TABLESPACE CONTAINERS FOR 4
-- IGNORE ROLLFORWARD CONTAINER OPERATIONS
USING (
    DEVICE '/dev/hdb1'              1000
    , DEVICE '/dev/hdb2'            1000
);
-- *****
-- ** start redirect restore
-- *****
RESTORE DATABASE TEST CONTINUE;
-- *****
-- ** end of file
-- *****

```

Usage notes:

- A RESTORE DATABASE command of the form db2 restore db <name> will perform a full database restore with a database image and will perform a table space restore operation of the table spaces found in a table space image. A RESTORE DATABASE command of the form db2 restore db <name> tablespace performs a table space restore of the table spaces found in the image. In addition, if a list of table spaces is provided with such a command, the explicitly listed table spaces are restored.
- Following the restore operation of an online backup, you must perform a roll-forward recovery.
- If a backup image is compressed, the DB2 database system detects this and automatically decompresses the data before restoring it. If a library is specified on the db2Restore API, it is used for decompressing the data. Otherwise, a check is made to see if a library is stored in the backup image and if it exists it is used. Finally, if there is not library stored in the backup image, the data cannot be decompressed and the restore operation fails.

- If the compression library is to be restored from a backup image (either explicitly by specifying the `COMPRESSION LIBRARY` option or implicitly by performing a normal restore of a compressed backup), the restore operation must be done on the same platform and operating system that the backup was taken on. If the platform the backup was taken on is not the same as the platform that the restore is being done on, the restore operation will fail, even if DB2 normally supports cross-platform restores involving the two systems.
- To restore log files from the backup image that contains them, the `LOGTARGET` option must be specified, providing the fully qualified and valid path that exists on the DB2 server. If those conditions are satisfied, the restore utility will write the log files from the image to the target path. If a `LOGTARGET` is specified during a restore of a backup image that does not include logs, the restore operation will return an error before attempting to restore any table space data. A restore operation will also fail with an error if an invalid, or read-only, `LOGTARGET` path is specified.
- If any log files exist in the `LOGTARGET` path at the time the `RESTORE DATABASE` command is issued, a warning prompt will be returned to the user. This warning will not be returned if `WITHOUT PROMPTING` is specified.
- During a restore operation where a `LOGTARGET` is specified, if any log file cannot be extracted, the restore operation will fail and return an error. If any of the log files being extracted from the backup image have the same name as an existing file in the `LOGTARGET` path, the restore operation will fail and an error will be returned. The restore database utility will not overwrite existing log files in the `LOGTARGET` directory.
- You can also restore only the saved log set from a backup image. To indicate that only the log files are to be restored, specify the `LOGS` option in addition to the `LOGTARGET` path. Specifying the `LOGS` option without a `LOGTARGET` path will result in an error. If any problem occurs while restoring log files in this mode of operation, the restore operation will terminate immediately and an error will be returned.
- During an automatic incremental restore operation, only the log files included in the target image of the restore operation will be retrieved from the backup image. Any log files included in intermediate images referenced during the incremental restore process will not be extracted from those intermediate backup images. During a manual incremental restore operation, the `LOGTARGET` path should only be specified with the final restore command to be issued.
- A backup targeted to be restored to another operating system or another DB2 database version must be an offline backup, and cannot be a delta or an incremental backup image. The same is true for backups to be restored to a later DB2 database version.

Related concepts:

- “Backup and restore operations between different operating systems and hardware platforms” in *Data Recovery and High Availability Guide and Reference*
- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

Related tasks:

- “Using restore” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “CREATE DATABASE ” on page 395
- “db2move - Database movement tool ” on page 157

REWIND TAPE

Rewinds tapes for backup and restore operations to streaming tape devices. This command is only supported on Windows operating systems.

Authorization:

One of the following:

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

Required connection:

None.

Command syntax:

►► REWIND TAPE ON device ◄◄

Command parameters:

ON device

Specifies a valid tape device name. The default value is `\\.\TAPE0`.

Related reference:

- “INITIALIZE TAPE ” on page 511
- “SET TAPE POSITION ” on page 723
- “REWIND TAPE command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

ROLLFORWARD DATABASE

Recovers a database by applying transactions recorded in the database log files. Invoked after a database or a table space backup image 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, the *logarchmeth1* or *logarchmeth2* database configuration parameters must be set to a value other than OFF) before the database can be recovered with rollforward recovery.

Scope:

In a partitioned database environment, this command can only be invoked from the catalog partition. A database or table space rollforward operation to a specified point in time affects all database partitions that are listed in the *db2nodes.cfg* file. A database or table space rollforward operation to the end of logs affects the database partitions that are specified. If no database partitions are specified, it affects all database partitions that are listed in the *db2nodes.cfg* file; if rollforward recovery is not needed on a particular partition, that partition is ignored.

For partitioned tables, you are also required to roll forward related table spaces to the same point in time. This applies to table spaces containing data partitions of a table. If a single table space contains a portion of a partitioned table, rolling forward to the end of the logs is still allowed.

Authorization:

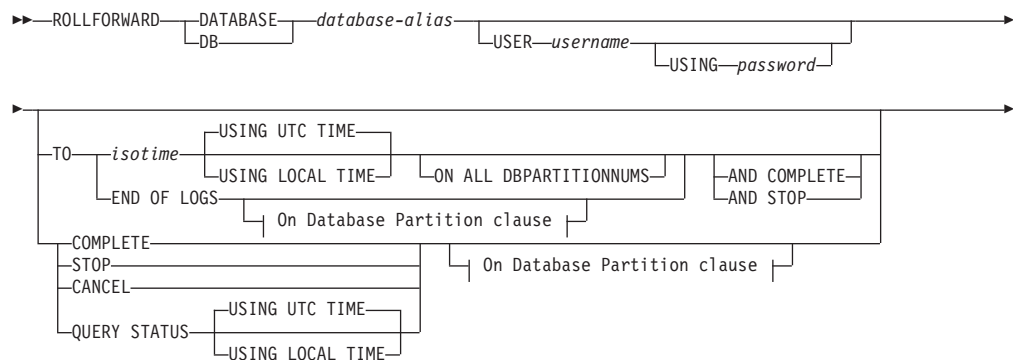
One of the following:

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

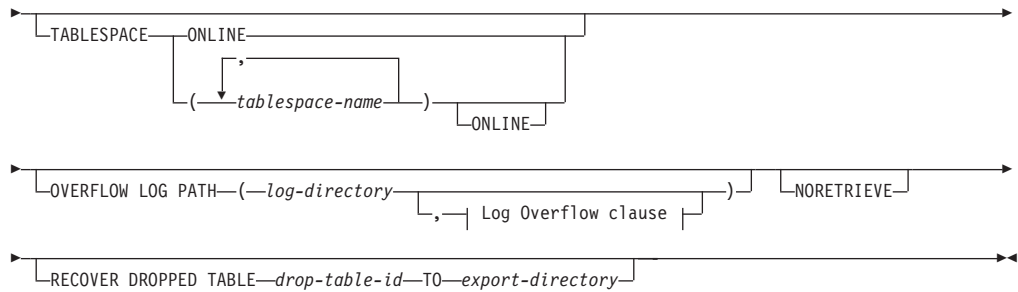
Required connection:

None. This command establishes a database connection.

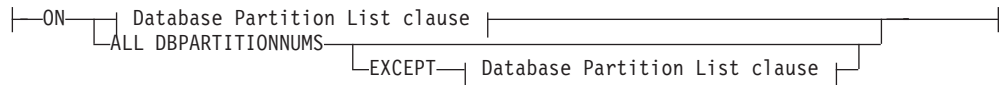
Command syntax:



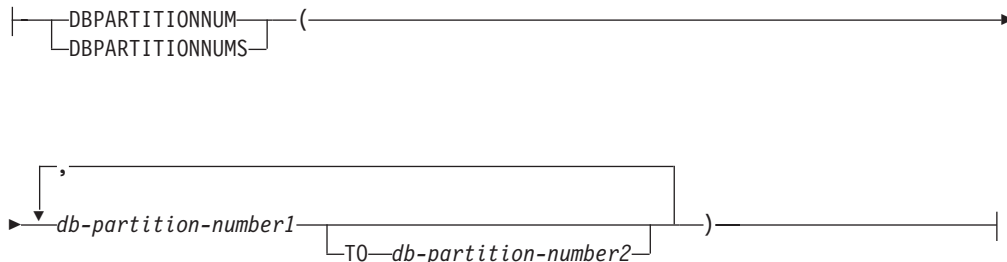
ROLLFORWARD DATABASE



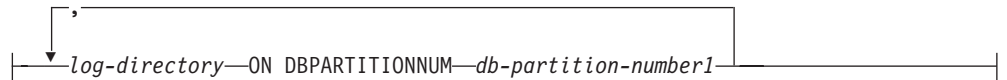
On Database Partition clause:



Database Partition List clause:



Log Overflow clause:



Command parameters:

DATABASE database-alias

The alias of the database that is to be rollforward recovered.

USER username

The user name under which the database is to be rollforward recovered.

USING password

The password used to authenticate the user name. If the password is omitted, you will be 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* (year, month, day, hour, minutes, seconds), expressed in Coordinated Universal Time (UTC, formerly known as GMT). UTC helps to avoid having the same time stamp associated

with different logs (because of a change in time associated with daylight savings time, for example). The time stamp in a backup image is based on the local time at which the backup operation started. The CURRENT TIMEZONE special register specifies the difference between UTC and local time at the application server. The difference is represented by a time duration (a decimal number in which the first two digits represent the number of hours, the next two digits represent the number of minutes, and the last two digits represent the number of seconds). Subtracting CURRENT TIMEZONE from a local time converts that local time to UTC.

USING LOCAL TIME

Allows you to rollforward to a point in time that is the server's local time rather than UTC time.

Notes:

1. If you specify a local time for rollforward, all messages returned to you will also be in local time. All times are converted on the server, and in partitioned database environments, on the catalog database partition.
2. The timestamp string is converted to UTC on the server, so the time is local to the server's time zone, not the client's. If the client is in one time zone and the server in another, the server's local time should be used. This is different from the local time option from the Control Center, which is local to the client.
3. If the timestamp string is close to the time change of the clock due to daylight savings, it is important to know if the stop time is before or after the clock change, and specify it correctly.
4. Subsequent ROLLFORWARD commands that cannot specify the USING LOCAL TIME clause will have all messages returned to you in local time if this option is specified.

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 DBPARTITIONNUMS

Specifies that transactions are to be rolled forward on all database partitions specified in the *db2nodes.cfg* file. This is the default if a database partition clause is not specified.

EXCEPT

Specifies that transactions are to be rolled forward on all database partitions specified in the *db2nodes.cfg* file, except those specified in the database partition list.

ON DBPARTITIONNUM / ON DBPARTITIONNUMS

Roll the database forward on a set of database partitions.

db-partition-number1

Specifies a database partition number in the database partition list.

db-partition-number2

Specifies the second database partition number, so that all database partitions from *db-partition-number1* up to and including *db-partition-number2* are included in the database partition list.

ROLLFORWARD DATABASE

COMPLETE / STOP

Stops the rolling forward of log records, and completes the rollforward recovery process by rolling back any incomplete transactions and turning off the rollforward 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`. When rolling table spaces forward to a point in time, the table spaces are placed in backup pending state.

CANCEL

Cancels the rollforward recovery operation. This puts the database or one or more table spaces on all database partitions on which forward recovery has been started in restore pending state:

- If a *database* rollforward operation is not in progress (that is, the database is in rollforward pending state), this option puts the database in restore pending state.
- If a *table space* rollforward operation 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 are put in restore pending state.
- If a table space rollforward operation *is* in progress (that is, at least one table space is in rollforward in progress state), all table spaces that are in rollforward in progress state are put in restore pending state. If a table space list is specified, it must include all table spaces that are in rollforward in progress state. All table spaces on the list are put in restore pending state.
- If rolling forward to a point in time, any table space name that is passed in is ignored, and all table spaces that are in rollforward in progress state are put in restore pending state.
- If rolling forward to the end of the logs with a table space list, only the table spaces listed are put in restore pending state.

This option cannot be used to cancel a rollforward operation *that is actually running*. It can only be used to cancel a rollforward operation that is in progress but not actually running at the time. A rollforward operation can be in progress but not running if:

- It terminated abnormally.
- The STOP option was not specified.
- An error caused it to fail. Some errors, such as rolling forward through a non-recoverable load operation, can put a table space into restore pending state.

Use this option with caution, and only if the rollforward operation that is in progress cannot be completed because some of the table spaces have been put in rollforward pending state or in restore pending state. When in doubt, use the LIST TABLESPACES command to identify the table spaces that are in rollforward in progress state, or in rollforward pending state.

QUERY STATUS

Lists the log files that the database manager has rolled forward, the next archive file required, and the time stamp (in UTC) of the last committed transaction since rollforward processing began. In a partitioned database environment, this status information is returned for each database partition. The information returned contains the following fields:

Database partition number**Rollforward status**

Status can be: database or table space rollforward pending, database or table space rollforward in progress, database or table space rollforward processing STOP, or not pending.

Next log file to be read

A string containing the name of the next required log file. In a partitioned database environment, use this information if the rollforward utility fails with a return code indicating that a log file is missing or that a log information mismatch has occurred.

Log files processed

A string containing the names of processed log files that are no longer needed for recovery, and that can be removed from the directory. If, for example, the oldest uncommitted transaction starts in log file x , the range of obsolete log files will not include x ; the range ends at $x - 1$.

Last committed transaction

A string containing a time stamp in ISO format ($yyyy-mm-dd-hh.mm.ss$) suffixed by either "UTC" or "Local" (see USING LOCAL TIME). This time stamp marks the last transaction committed after the completion of rollforward recovery. The time stamp applies to the database. For table space rollforward recovery, it is the time stamp of the last transaction committed to the database.

QUERY STATUS is the default value if the TO, STOP, COMPLETE, or CANCEL clauses are omitted. If TO, STOP, or COMPLETE was specified, status information is displayed if the command has completed successfully. If individual table spaces are specified, they are ignored; the status request does not apply only to specified table spaces.

TABLESPACE

This keyword is specified for table space-level rollforward recovery.

tablespace-name

Mandatory for table space-level rollforward recovery to a point in time. Allows a subset of table spaces to be specified for rollforward recovery to the end of the logs. In a partitioned database environment, each table space in the list does not have to exist at each database partition that is rolling forward. If it *does* exist, it must be in the correct state.

For partitioned tables, point in time roll-forward of a table space containing any piece of a partitioned table must also roll-forward all of the other table spaces in which that table resides to the same point in time. Roll-forward to the end of the logs for a single table space containing a piece of a partitioned table is still allowed.

If a partitioned table has any attached or detached data partitions, then PIT rollforward must include all table spaces for these data partitions as well. To determine if a partitioned table has any attached, detached, or dropped data partitions, query the Status field of the SYSDATAPARTITIONS catalog table.

Because a partitioned table can reside in multiple table spaces, it will generally be necessary to roll forward multiple table spaces. Data that is recovered via dropped table recovery is written to the export directory specified in the ROLLFORWARD DATABASE command. It is possible to

ROLLFORWARD DATABASE

roll forward all table spaces in one command, or do repeated roll forward operations for subsets of the table spaces involved. If the ROLLFORWARD DATABASE command is done for one or a few table spaces, then all data from the table that resided in those table spaces will be recovered. A warning will be written to the notify log if the ROLLFORWARD DATABASE command did not specify the full set of the table spaces necessary to recover all the data for the table. Allowing rollforward of a subset of the table spaces makes it easier to deal with cases where there is more data to be recovered than can fit into a single export directory.

ONLINE

This keyword is specified to allow table space-level rollforward recovery to be done online. This means that other agents are allowed to connect while rollforward recovery is in progress.

OVERFLOW LOG PATH **log-directory**

Specifies an alternate log path to be searched for archived logs during recovery. Use this parameter if log files were moved to a location other than that specified by the *logpath* database configuration parameter. In a partitioned database environment, this is the (fully qualified) default overflow log path *for all database partitions*. A relative overflow log path can be specified for single-partition databases. The OVERFLOW LOG PATH command parameter will overwrite the value (if any) of the database configuration parameter OVERFLOWLOGPATH.

log-directory ON DBPARTITIONNUM

In a partitioned database environment, allows a different log path to override the default overflow log path for a specific database partition.

NORETRIEVE

Allows you to control which log files are to be rolled forward on the standby machine by allowing you to disable the retrieval of archived logs. The benefits of this are:

- By controlling the logfiles to be rolled forward, you can ensure that the standby machine is X hours behind the production machine, to avoid affecting both the systems.
- If the standby system does not have access to archive (eg. if TSM is the archive, it only allows the original machine to retrieve the files)
- It might also be possible that while the production system is archiving a file, the standby system is retrieving the same file, and it might then get an incomplete log file. Noretrieve would solve this problem.

RECOVER DROPPED TABLE **drop-table-id**

Recovers a dropped table during the rollforward operation. The table ID can be obtained using the LIST HISTORY command. For partitioned tables, the drop-table-id identifies the table as a whole, so that all data partitions of the table can be recovered in a single roll-forward command.

TO **export-directory**

Specifies a directory to which files containing the table data are to be written. The directory must be accessible to all database partitions.

Examples:

Example 1

The ROLLFORWARD DATABASE 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
```

Although the two are equivalent, it is recommended that such operations be done in two steps. It is important to verify that the rollforward operation has progressed as expected, before stopping it and possibly missing logs. This is especially important if a bad log is found during rollforward recovery, and the bad log is interpreted to mean the “end of logs”. In such cases, an undamaged backup copy of that log could be used to continue the rollforward operation through more logs. However if the rollforward AND STOP option is used, and the rollforward encounters an error, the error will be returned to you. In this case, the only way to force the rollforward to stop and come online despite the error (i.e. to come online at that point in the logs before the error) is to issue the rollforward STOP command.

Example 2

Roll forward to the end of the logs (two table spaces have been restored):

```
db2 rollforward db sample to end of logs
db2 rollforward db sample to end of logs and stop
```

These two statements are equivalent. Neither AND STOP or AND COMPLETE is needed for table space rollforward recovery to the end of the logs. Table space names are not required. If not specified, all table spaces requiring rollforward recovery will be included. If only a subset of these table spaces is to be rolled forward, their names must be specified.

Example 3

After three table spaces have been restored, roll one forward to the end of the 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 and stop
tablespace(TBS2, TBS3) online
```

Two rollforward operations cannot be run concurrently. The second command can only be invoked after the first rollforward operation completes successfully.

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 and stop
overflow log path (/logs)
```

Example 5 (partitioned database environments)

There are three database partitions: 0, 1, and 2. Table space TBS1 is defined on all database partitions, and table space TBS2 is defined on database partitions 0 and 2.

ROLLFORWARD DATABASE

After restoring the database on database partition 1, and TBS1 on database partitions 0 and 2, roll the database forward on database partition 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 database partition(s) 0 and 2.").

```
db2 rollforward db sample to end of logs
```

This rolls TBS1 forward on database partitions 0 and 2. The clause TABLESPACE(TBS1) is optional in this case.

Example 6 (partitioned database environments)

After restoring table space TBS1 on database partitions 0 and 2 only, roll TBS1 forward on database partitions 0 and 2:

```
db2 rollforward db sample to end of logs
```

Database partition 1 is ignored.

```
db2 rollforward db sample to end of logs tablespace(TBS1)
```

This fails, because TBS1 is not ready for rollforward recovery on database partition 1. Reports SQL4906N.

```
db2 rollforward db sample to end of logs on dbpartitionnums (0, 2)
tablespace(TBS1)
```

This completes successfully.

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
tablespace(TBS1)
```

This fails, because TBS1 is not ready for rollforward recovery on database partition 1; all pieces must be rolled forward together. With table space rollforward to a point in time, the database partition clause is not accepted. The rollforward operation must take place on all the database partitions on which the table space resides.

After restoring TBS1 on database partition 1:

```
db2 rollforward db sample to 1998-04-03-14.21.56 and stop
tablespace(TBS1)
```

This completes successfully.

Example 7 (partitioned database environment)

After restoring a table space on all database partitions, roll forward to point in time 2, but do not specify AND STOP. The rollforward operation is still in progress. Cancel and roll forward to point in time 1:

```
db2 rollforward db sample to pit2 tablespace(TBS1)
db2 rollforward db sample cancel tablespace(TBS1)
```

```
** restore TBS1 on all database partitions **
```

```
db2 rollforward db sample to pit1 tablespace(TBS1)
db2 rollforward db sample stop tablespace(TBS1)
```

Example 8 (partitioned database environments)

Rollforward recover a table space that resides on eight database partitions (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 database partitions on which the table space resides do not have to be specified. The utility defaults to the `db2nodes.cfg` file.

Example 9 (partitioned database environment)

Rollforward recover six small table spaces that reside on a single-partition database partition group (on database partition 6):

```
db2 rollforward database dwtest to end of logs on dbpartitionnum (6)
tablespace(tsstore, tssbuyer, tsstime, tsswhse, tsslscat, tssvendor)
```

This operation to the end of logs (not point in time) completes successfully.

Usage notes:

If restoring from an image that was created during an online backup operation, the specified point in time for the rollforward operation must be later than the time at which the online backup operation completed. If the rollforward operation is stopped before it passes this point, the database is left in rollforward pending state. If a table space is in the process of being rolled forward, it is left in rollforward in progress state.

If one or more table spaces is being rolled forward to a point in time, the rollforward operation must continue at least to the minimum recovery time, which is the last update to the system catalogs for this table space or its tables. The minimum recovery time (in Coordinated Universal Time, or UTC) for a table space can be retrieved using the `LIST TABLESPACES SHOW DETAIL` command.

Rolling databases forward might require a load recovery using tape devices. If prompted for another tape, you 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. Take all affected tablespaces offline, but continue rollforward processing.

If the rollforward utility cannot find the next log that it needs, the log name is returned in the `SQLCA`, and rollforward recovery stops. If no more logs are available, use the `STOP` option to terminate rollforward recovery. Incomplete transactions are rolled back to ensure that the database or table space is left in a consistent state.

Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword `NODE` can be substituted for `DBPARTITIONNUM`.
- The keyword `NODES` can be substituted for `DBPARTITIONNUMS`.
- Point in time rollforward is not supported with pre-V9.1 clients due to V9.1 support for partitioned tables.

ROLLFORWARD DATABASE

Related concepts:

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

Related tasks:

- “Using rollforward” in *Data Recovery and High Availability Guide and Reference*

RUNCMD

Executes a specified command from the CLP interactive mode command history.

Scope

This command can only be run within CLP interactive mode. Specifically, it cannot be run from the CLP command mode or the CLP batch mode.

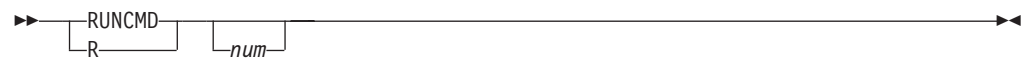
Authorization:

None

Required connection:

The required connection will depend on the command being executed.

Command syntax:



Command parameters:

num If *num* is positive, executes the command corresponding to *num* in the command history. If *num* is negative, executes the command corresponding to *num*, counting backwards from the most recent command in the command history. Zero is not a valid value for *num*. If this parameter is not specified, executes the most recently run command. (This is equivalent to specifying a value of -1 for *num*).

Usage notes:

1. Typically, you would execute the HISTORY command to see a list of recently executed commands and then execute the RUNCMD to execute a command from this list.
2. The RUNCMD command is not recorded in the command history, but the command executed by the RUNCMD command is recorded in the command history.

Related reference:

- "EDIT " on page 432
- "HISTORY " on page 493

RUNSTATS

Updates statistics about the characteristics of a table and/or associated indexes, or statistical views. 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.

For a table, this utility should be called when the table has had many updates, or after reorganizing the table. For a statistical view, this utility should be called when changes to underlying tables have substantially affected the rows returned by the view. The view must have been previously enabled for use in query optimization using the **ALTER VIEW** command.

Scope:

This command can be issued from any database partition in the `db2nodes.cfg` file. It can be used to update the catalogs on the catalog database partition.

For tables, this command collects statistics for a table on the database partition from which it is invoked. If the table does not exist on that database partition, the first database partition in the database partition group is selected.

For views, this command collects statistics using data from tables on all participating database partitions.

Authorization:

For tables, one of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the table
- LOAD authority

You do not need any explicit privilege to use this command on any declared global temporary table that exists within its connection.

For statistical views, one of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the statistical view

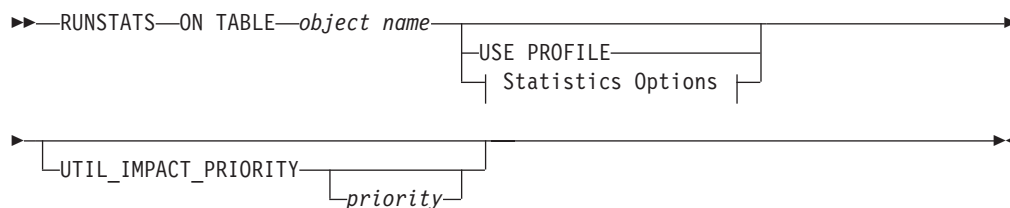
In addition, you need to have appropriate privileges to access rows from the statistical view. Specifically, for each table, statistical view or nickname referenced in the statistical view definition, the user must have one of the following privileges:

- *sysadm* or *dbadm*
- CONTROL
- SELECT

Required connection:

Database

Command syntax:



Statistics Options:

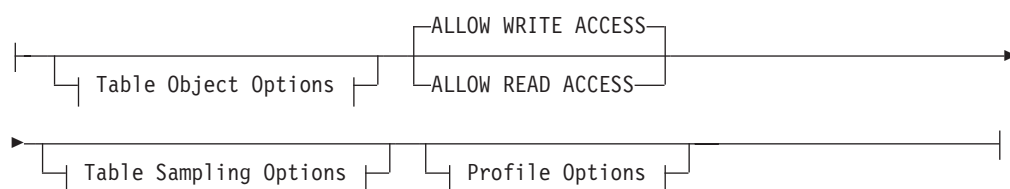


Table Object Options:

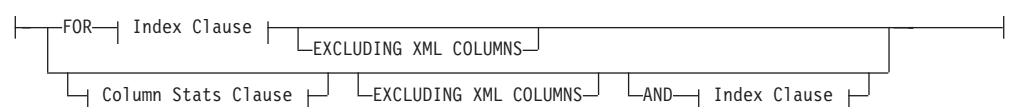
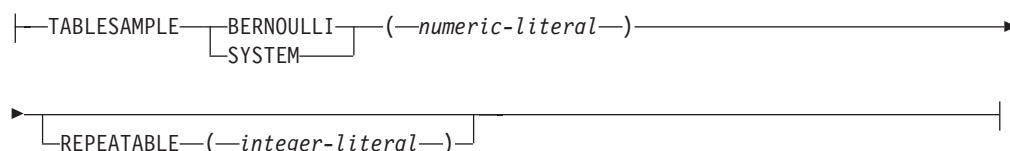
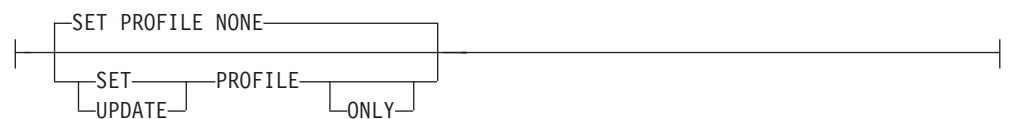


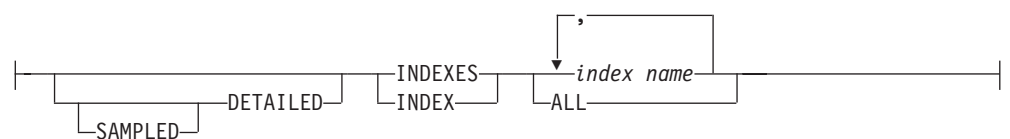
Table Sampling Options:



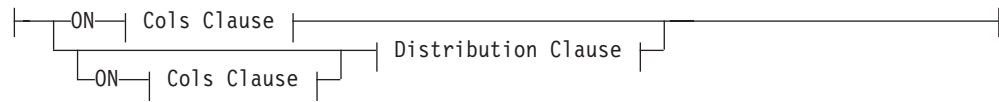
Profile Options:



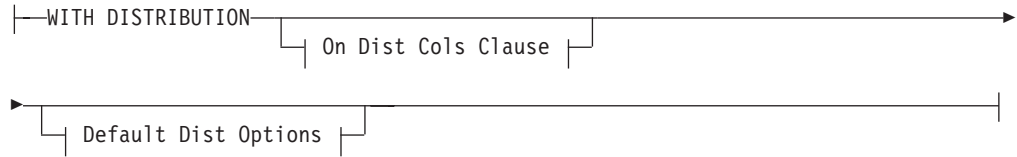
Index Clause:



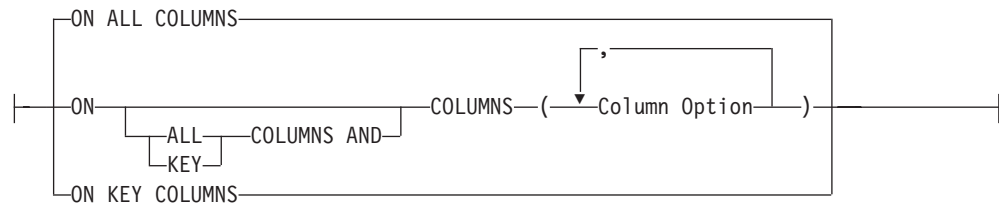
Column Stats Clause:



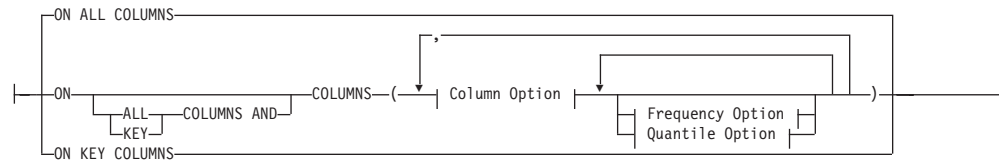
Distribution Clause:



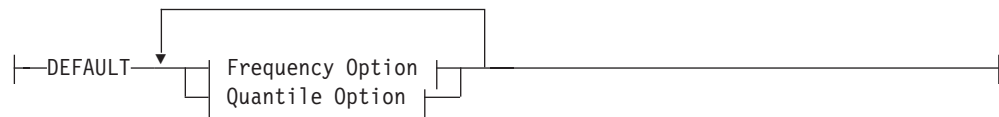
On Cols Clause:



On Dist Cols Clause:



Default Dist Option:



Frequency Option:



Quantile Option:



Column Option:

WITH DISTRIBUTION clause. Those columns specified as part of the WITH DISTRIBUTION clause will also have basic and distribution statistics collected.

If the WITH DISTRIBUTION ON ALL COLUMNS is specified both basic statistics and distribution statistics are collected for all eligible columns. Anything specified in the on-cols-clause is redundant and therefore not necessary.

ON COLUMNS

This clause allows the user to specify a list of columns for which to collect statistics. If you specify group of columns, the number of distinct values for the group will be collected. When you run **RUNSTATS** on a table without gathering index statistics, and specify a subset of columns for which statistics are to be gathered, then:

1. Statistics for columns not specified in the **RUNSTATS** command but which are the first column in an index are NOT reset.
2. Statistics for all other columns not specified in the **RUNSTATS** command are reset.

This clause can be used in the on-cols-clause and the on-dist-cols-clause. Collecting distribution statistics for a group of columns is not currently supported.

If XML type columns are specified in a column group, the XML type columns will be ignored for the purpose of collecting distinct values for the group. However, basic XML column statistics will be collected for the XML type columns in the column group.

EXCLUDING XML COLUMNS

This clause allows you to omit all XML type columns from statistics collection. This clause facilitates the collection of statistics on non-XML columns because the inclusion of XML data can require greater system resources. The EXCLUDING XML COLUMNS clause takes precedence over other clauses that specify XML columns for statistics collection. For example, if you use the EXCLUDING XML COLUMNS clause, and you also specify XML type columns with the ON COLUMNS clause or you use the ON ALL COLUMNS clause, all XML type columns will be ignored during statistics collection.

ON KEY COLUMNS

Instead of listing specific columns, you can choose to collect statistics on columns that make up all the indexes defined on the table. It is assumed here that critical columns in queries are also those used to create indexes on the table. If there are no indexes on the table, it is as good as an empty list and no column statistics will be collected. It can be used in the on-cols-clause or the on-dist-cols-clause. It is redundant in the on-cols-clause if specified in both clauses since the WITH DISTRIBUTION clause is used to specify collection of both basic and distribution statistics. XML type columns are by definition not a key column and will not be included for statistics collection by the ON KEY COLUMNS clause. This option cannot be used for views.

column-name

Name of a column in the table or statistical view. If you specify the name of an ineligible column for statistics collection, such as a non-existent column or a mistyped column name, error (-205) is returned. Two lists of columns can be specified, one without distribution and one with distribution. If the column is specified in the list that is not associated with the WITH DISTRIBUTION clause only basic column statistics will be collected.

If the column appears in both lists, distribution statistics will be collected (unless NUM_FREQVALUES and NUM_QUANTILES are set to zero).

NUM_FREQVALUES

Defines the maximum number of frequency values to collect. It can be specified for an individual column in the ON COLUMNS clause. If the value is not specified for an individual column, the frequency limit value will be picked up from that specified in the DEFAULT clause. If it is not specified there either, the maximum number of frequency values to be collected will be what is set in the NUM_FREQVALUES database configuration parameter.

NUM_QUANTILES

Defines the maximum number of distribution quantile values to collect. It can be specified for an individual column in the ON COLUMNS clause. If the value is not specified for an individual column, the quantile limit value will be picked up from that specified in the DEFAULT clause. If it is not specified there either, the maximum number of quantile values to be collected will be what is set in the NUM_QUANTILES database configuration parameter.

WITH DISTRIBUTION

This clause specifies that both basic statistics and distribution statistics are to be collected on the columns. If the ON COLUMNS clause is not specified, distribution statistics are collected on all the columns of the table or statistical view (excluding columns that are ineligible such as CLOB and LONG VARCHAR). If the ON COLUMNS clause is specified, distribution statistics are collected only on the column list provided (excluding those ineligible for statistics collection). If the clause is not specified, only basic statistics are collected.

Collection of distribution statistics on column groups is currently not supported; distribution statistics will not be collected when column groups are specified in the WITH DISTRIBUTION ON COLUMNS clause.

DEFAULT

If NUM_FREQVALUES or NUM_QUANTILES are specified, these values will be used to determine the maximum number of frequency and quantile statistics to be collected for the columns, if these are not specified for individual columns in the ON COLUMNS clause. If the DEFAULT clause is not specified, the values used will be those in the corresponding database configuration parameters.

LIKE STATISTICS

When this option is specified additional column statistics are collected. These statistics are the SUB_COUNT and the SUB_DELIM_LENGTH statistics in SYSSTAT.COLUMNS. They are collected for string columns only and they are used by the query optimizer to improve the selectivity estimates for predicates of the type "column LIKE '%xyz'" and "column LIKE '%xyz%'"

ALLOW WRITE ACCESS

Specifies that other users can read from and write to the table(s) while statistics are calculated. For statistical views, these are the base tables referenced in the view definition.

The ALLOW WRITE ACCESS option is not recommended for tables that will have a lot of inserts, updates or deletes occurring concurrently. The RUNSTATS command first performs table statistics and then performs index statistics. Changes in the table's state between the time that the table and index statistics are collected might result in inconsistencies. Although having up-to-date statistics is important for the optimization of queries, it

is also important to have consistent statistics. Therefore, statistics should be collected at a time when inserts, updates or deletes are at a minimum.

ALLOW READ ACCESS

Specifies that other users can have read-only access to the table(s) while statistics are calculated. For statistical views, these are the base tables referenced in the view definition.

TABLESAMPLE BERNOULLI

This option allows **RUNSTATS** to collect statistics on a sample of the rows from the table or statistical view. Bernoulli sampling considers each row individually, including that row with probability P/100 (where P is the value of numeric-literal) and excluding it with probability 1-P/100. Thus, if the numeric-literal were evaluated to be the value 10, representing a 10 percent sample, each row would be included with probability 0.1 and be excluded with probability 0.9. Unless the optional REPEATABLE clause is specified, each execution of **RUNSTATS** will usually yield a different such sample of the table. All data pages will be retrieved through a table scan but only the percentage of rows as specified through the numeric-literal parameter will be used for the statistics collection.

TABLESAMPLE SYSTEM

This option allows **RUNSTATS** to collect statistics on a sample of the data pages from the table(s). System sampling considers each page individually, including that page with probability P/100 (where P is the value of numeric-literal) and excluding it with probability 1-P/100. Unless the optional REPEATABLE clause is specified, each execution of **RUNSTATS** will usually yield a different such sample of the table. The size of the sample is controlled by the numeric-literal parameter in parentheses, representing an approximate percentage P of the table to be returned. Only a percentage of the data pages as specified through the numeric-literal parameter will be retrieved and used for the statistics collection. On statistical views, system sampling is restricted to a specific class of views. These are views that either access a single base table or nickname, or that access multiple bases tables that are joined via referential-integrity relationships. In either case, there must not be any local predicates in the view definition. If system sampling is specified on a view that cannot support such sampling, an SQL20288N error is raised.

REPEATABLE (*integer-literal*)

Adding the REPEATABLE clause to the TABLESAMPLE clause ensures that repeated executions of **RUNSTATS** return the same sample. The *integer-literal* parameter is a non-negative integer representing the seed to be used in sampling. Passing a negative seed will result in an error (SQL1197N). The sample set might still vary between repeatable **RUNSTATS** invocations if activity against the table or statistical view resulted in changes to the table or statistical view data since the last time TABLESAMPLE REPEATABLE was run. Also, the method by which the sample was obtained as specified by the bernoulli or system keyword, must also be the same to ensure consistent results.

numeric-literal

The numeric-literal parameter specifies the size of the sample to be obtained, as a percentage P. This value must be a positive number that is less than or equal to 100, and can be between 1 and 0. For example, a value of 0.01 represents one one-hundredth of a percent, such that 1 row in 10,000 would be sampled, on average. A value of 0 or 100 will be treated by the DB2 database system as if sampling was not specified, regardless of

whether TABLESAMPLE BERNOULLI or TABLESAMPLE SYSTEM is specified. A value greater than 100 or less than 0 will be treated by DB2 as an error (SQL1197N).

SET PROFILE NONE

Specifies that no statistics profile will be set for this **RUNSTATS** invocation.

SET PROFILE

Allows **RUNSTATS** to generate and store a specific statistics profile in the system catalog tables and executes the **RUNSTATS** command options to gather statistics.

SET PROFILE ONLY

Allows **RUNSTATS** to generate and store a specific statistics profile in the system catalog tables without running the **RUNSTATS** command options.

UPDATE PROFILE

Allows **RUNSTATS** to modify an existing statistics profile in the system catalog tables, and runs the **RUNSTATS command options of the updated statistics profile to gather statistics**.

UPDATE PROFILE ONLY

Allows **RUNSTATS** to modify an existing statistics profile in the system catalog tables without running the **RUNSTATS** command options of the updated statistics profile.

UTIL_IMPACT_PRIORITY *priority*

Specifies that **RUNSTATS** will be throttled at the level specified by *priority*. *priority* is a number in the range of 1 to 100, with 100 representing the highest priority and 1 representing the lowest. The priority specifies the amount of throttling to which the utility is subjected. All utilities at the same priority undergo the same amount of throttling, and utilities at lower priorities are throttled more than those at higher priorities. If *priority* is not specified, the **RUNSTATS** will have the default priority of 50. Omitting the **UTIL_IMPACT_PRIORITY** keyword will invoke the **RUNSTATS** utility without throttling support. If the **UTIL_IMPACT_PRIORITY** keyword is specified, but the *util_impact_lim* configuration parameter is set to 100, then the utility will run unthrottled. This option cannot be used for views.

In a partitioned database, when used on tables, the **RUNSTATS** command collects the statistics on only a single database partition. If the database partition from which the **RUNSTATS** command is executed has a partition of the table, then the command executes on that database partition. Otherwise, the command executes on the first database partition in the database partition group across which the table is partitioned.

Usage Notes:

1. When there are detached partitions on a partitioned table, index keys that still belong to detached data partitions which require cleanup will not be counted as part of the keys in the statistics. These keys are not counted because they are invisible and no longer part of the table. They will eventually get removed from the index by asynchronous index cleanup. As a result, statistics collected before asynchronous index cleanup is run will be misleading. If the **RUNSTATS** command is issued before asynchronous index cleanup completes, it will likely generate a false alarm for index reorganization or index cleanup based on the inaccurate statistics. Once asynchronous index

cleanup is run, all the index keys that still belong to detached data partitions which require cleanup will be removed and this may eliminate the need for index reorganization.

For partitioned tables, you are encouraged to issue the **RUNSTATS** command after an asynchronous index cleanup has completed in order to generate accurate index statistics in the presence of detached data partitions. To determine whether or not there are detached data partitions in the table, you can check the status field in the `SYSDATAPARTITIONS` table and look for the value I (index cleanup) or D (detached with dependant MQT).

2. It is recommended to run the **RUNSTATS** command:
 - On tables that have been modified considerably (for example, if a large number of updates have been made, or if a significant amount of data has been inserted or deleted or if **LOAD** has been done without the statistics option during **LOAD**).
 - On tables that have been reorganized (using **REORG**, **REDISTRIBUTE DATABASE PARTITION GROUP**).
 - On tables which have been row compressed.
 - When a new index has been created.
 - Before binding applications whose performance is critical.
 - When the prefetch quantity is changed.
 - On statistical views whose underlying tables have been modified substantially so as to change the rows that are returned by the view.
 - After **LOAD** has been executed with the **STATISTICS** option, use the **RUNSTATS** utility to collect statistics on XML columns. Statistics for XML columns are never collected during **LOAD**, even when **LOAD** is executed with the **STATISTICS** option. When **RUNSTATS** is used to collect statistics for XML columns only, existing statistics for non-XML columns that have been collected by **LOAD** or a previous execution of the **RUNSTATS** utility are retained. In the case where statistics on some XML columns have been collected previously, the previously collected statistics for an XML column will either be dropped if no statistics on that XML column are collected by the current command, or be replaced if statistics on that XML column are collected by the current command.
3. The options chosen must depend on the specific table and the application. In general:
 - If the table is a very critical table in critical queries, is relatively small, or does not change too much and there is not too much activity on the system itself, it might be worth spending the effort on collecting statistics in as much detail as possible.
 - If the time to collect statistics is limited, if the table is relatively large, or if the table is updated frequently, it might be beneficial to execute **RUNSTATS** limited to the set of columns that are used in predicates. This way, you will be able to execute the **RUNSTATS** command more often.
 - If time to collect statistics is very limited and the effort to tailor the **RUNSTATS** command on a table by table basis is a major issue, consider collecting statistics for the "KEY" columns only. It is assumed that the index contains the set of columns that are critical to the table and are most likely to appear in predicates.
 - If time to collect statistics is very limited and table statistics are to be gathered, consider using the **TABLESAMPLE** option to collect statistics on a subset of the table data.

- If there are many indexes on the table and DETAILED (extended) information on the indexes might improve access plans, consider the SAMPLED option to reduce the time it takes to collect statistics. Regardless of whether you use the SAMPLED option, collecting detailed statistics on indexes is time consuming. Do not collect these statistics unless you are sure that they will be useful for your queries.
 - If there is skew in certain columns and predicates of the type "column = constant", it might be beneficial to specify a larger NUM_FREQVALUES value for that column
 - Collect distribution statistics for all columns that are used in equality predicates and for which the distribution of values might be skewed.
 - For columns that have range predicates (for example "column >= constant", "column BETWEEN constant1 AND constant2") or of the type "column LIKE '%xyz'", it might be beneficial to specify a larger NUM_QUANTILES value.
 - If storage space is a concern and one cannot afford too much time on collecting statistics, do not specify high NUM_FREQVALUES or NUM_QUANTILES values for columns that are not used in predicates.
 - 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.
 - If statistics for XML columns in the table are not required, the EXCLUDING XML COLUMNS option can be used to exclude all XML columns. This option takes precedence over all other clauses that specify XML columns for statistics collection.
4. After the command is run note the following:
 - 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.
 - Executing the command on portions of the table could result in inconsistencies as a result of activity on the table since the command was last issued. In this case a warning message is returned. Issuing **RUNSTATS** on the table only might make table and index level statistics inconsistent. For example, you might collect index level statistics on a table and later delete a significant number of rows from the table. If you then issue **RUNSTATS** on the table only, the table cardinality might be less than FIRSTKEYCARD, which is an inconsistency. In the same way, if you collect statistics on a new index when you create it, the table level statistics might be inconsistent.
 5. The **RUNSTATS** command will drop previously collected distribution statistics if table statistics are requested. For example, **RUNSTATS ON TABLE**, or **RUNSTATS ON TABLE ... AND INDEXES ALL** will cause previously collected distribution statistics to be dropped. If the command is run on indexes only then previously collected distribution statistics are retained. For example, **RUNSTATS ON TABLE ... FOR INDEXES ALL** will cause the previously collected distribution statistics to be retained. If the **RUNSTATS** command is run on XML columns only, then previously collected basic column statistics and distribution statistics are retained. In the case where statistics on some XML columns have been collected previously, the previously collected statistics for an XML column will either be dropped if no statistics on that XML column are collected by the current command, or be replaced if statistics on that XML column are collected by the current command.

RUNSTATS

6. For range-clustered tables, there is a special system-generated index in the catalog tables which represents the range ordering property of range-clustered tables. When statistics are collected on this type of table, if the table is to be included as part of the statistics collection, statistics will also be collected for the system-generated index. The statistics reflect the fast access of the range lookups by representing the index as a two-level index with as many pages as the base data table, and having the base data clustered perfectly along the index order.
7. In the `on-dist-cols` clause of the command syntax, the `Frequency Option` and `Quantile Option` parameters are currently not supported for `Column GROUPS`. These options are supported for single columns.
8. There are three prefetch statistics that cannot be computed when working in DMS mode. When looking at the index statistics in the index catalogs, you will see a -1 value for the following statistics:
 - `AVERAGE_SEQUENCE_FETCH_PAGES`
 - `AVERAGE_SEQUENCE_FETCH_GAP`
 - `AVERAGE_RANDOM_FETCH_PAGES`
9. Runstats sampling through `TABLESAMPLE` only occurs with table data pages and not index pages. When index statistics as well as sampling is requested, all the index pages are scanned for statistics collection. It is only in the collection of table statistics where `TABLESAMPLE` is applicable. However, a more efficient collection of detailed index statistics is available through the `SAMPLED DETAILED` option. This is a different method of sampling than that employed by `TABLESAMPLE` and only applies to the detailed set of index statistics.
10. A statistics profile can be set or updated for the table or statistical view specified in the **RUNSTATS** command, by using the `set profile` or `update profile` options. The statistics profile is stored in a visible string format, which represents the **RUNSTATS** command, in the `STATISTICS_PROFILE` column of the `SYSIBM.SYSTABLES` system catalog table.
11. Statistics collection on XML type columns is governed by two DB2 database system registry values: `DB2_XML_RUNSTATS_PATHID_K` and `DB2_XML_RUNSTATS_PATHVALUE_K`. These two parameters are similar to the `NUM_FREQVALUES` parameter in that they specify the number of frequency values to collect. If not set, a default of 200 will be used for both parameters.
12. **RUNSTATS** acquires an IX table lock on `SYSTABLES` and a U lock on the row for the table on which stats are being gathered at the beginning of **RUNSTATS**. Operations can still read from `SYSTABLES` including the row with the U lock. Write operations are also possible, providing they do not occur against the row with the U lock. However, another reader or writer will not be able acquire an S lock on `SYSTABLES` because of **RUNSTATS'** IX lock.

Examples:

1. Collect statistics on the table only, on all columns without distribution statistics:

```
RUNSTATS ON TABLE db2user.employee
```
2. Collect statistics on the table only, on columns `empid` and `empname` with distribution statistics:

```
RUNSTATS ON TABLE db2user.employee
WITH DISTRIBUTION ON COLUMNS (empid, empname)
```
3. Collect statistics on the table only, on all columns with distribution statistics using a specified number of frequency limit for the table while picking the `NUM_QUANTILES` from the configuration setting:

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION DEFAULT
NUM_FREQVALUES 50
```

4. Collect statistics on a set of indexes:

```
RUNSTATS ON TABLE db2user.employee FOR INDEXES
db2user.emp11, db2user.emp12
```

5. Collect basic statistics on all indexes only:

```
RUNSTATS ON TABLE db2user.employee FOR INDEXES ALL
```

6. Collect basic statistics on the table and all indexes using sampling for the detailed index statistics collection:

```
RUNSTATS ON TABLE db2user.employee AND SAMPLED DETAILED INDEXES ALL
```

7. Collect statistics on table, with distribution statistics on columns empid, empname and empdept and the two indexes Xempid and Xempname. Distribution statistics limits are set individually for empdept, while the other two columns use a common default:

```
RUNSTATS ON TABLE db2user.employee
WITH DISTRIBUTION ON COLUMNS (empid, empname, empdept NUM_FREQVALUES
50 NUM_QUANTILES 100)
DEFAULT NUM_FREQVALUES 5 NUM_QUANTILES 10
AND INDEXES db2user.Xempid, db2user.Xempname
```

8. Collect statistics on all columns used in indexes and on all indexes:

```
RUNSTATS ON TABLE db2user.employee ON KEY COLUMNS AND INDEXES ALL
```

9. Collect statistics on all indexes and all columns without distribution except for one column. Consider T1 containing columns c1, c2, ..., c8

```
RUNSTATS ON TABLE db2user.T1
WITH DISTRIBUTION ON COLUMNS (c1, c2, c3 NUM_FREQVALUES 20
NUM_QUANTILES 40, c4, c5, c6, c7, c8)
DEFAULT NUM_FREQVALUES 0, NUM_QUANTILES 0
AND INDEXES ALL
```

```
RUNSTATS ON TABLE db2user.T1
WITH DISTRIBUTION ON COLUMNS (c3 NUM_FREQVALUES 20 NUM_QUANTILES 40)
AND INDEXES ALL
```

10. Collect statistics on table T1 for the individual columns c1 and c5 as well as on the column combinations (c2, c3) and (c2, c4). Multi-column cardinality is very useful to the query optimizer when it estimates filter factors for predicates on columns in which the data is correlated.

```
RUNSTATS ON TABLE db2user.T1 ON COLUMNS (c1, (c2, c3),
(c2, c4), c5)
```

11. Collect statistics on table T1 for the individual columns c1 and c2. For column c1 also collect the LIKE predicate statistics.

```
RUNSTATS ON TABLE db2user.T1 ON COLUMNS (c1 LIKE STATISTICS, c2)
```

12. Register a statistics profile to collect statistics on the table only, on all columns with distribution statistics using a specified number of frequency limit for the table while picking the NUM_QUANTILES from the configuration setting. The command also updates the statistics as specified.

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION DEFAULT
NUM_FREQVALUES 50 SET PROFILE
```

13. Register a statistics profile to collect statistics on the table only, on all columns with distribution statistics using a specified number of frequency limit for the table while picking the NUM_QUANTILES from the configuration setting. Statistics are not collected.

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION
DEFAULT NUM_FREQVALUES 50 SET PROFILE ONLY
```

RUNSTATS

14. Modify the previously registered statistics profile by changing the `NUM_FREQVALUES` value from 50 to 30. The command also updates the statistics as specified.

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION
      DEFAULT NUM_FREQVALUES 30 UPDATE PROFILE
```

15. Modify the previously registered statistics profile by changing the `NUM_FREQVALUES` value from 50 to 30. Statistics are not collected.

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION
      DEFAULT NUM_FREQVALUES 30 UPDATE PROFILE ONLY
```

16. Modify the previously registered statistics profile by adding column `empl_address` and column group (`empl_title`, `empl_salary`) options. The command also updates the statistics as specified.

```
RUNSTATS ON TABLE db2user.employee
      ON COLUMNS (empl_address, (empl_title, empl_salary)) UPDATE
      PROFILE
```

17. Modify the previously registered statistics profile by adding column `empl_address` and column group (`empl_title`, `empl_salary`) options. Statistics are not collected.

```
RUNSTATS ON TABLE db2user.employee
      ON COLUMNS (empl_address, (empl_title, empl_salary)) UPDATE
      PROFILE ONLY
```

18. Collect statistics on a table using the options recorded in the statistics profile for that table:

```
RUNSTATS ON TABLE db2user.employee USE PROFILE
```

19. Query the **RUNSTATS** command options corresponding to the previously registered statistics profile stored in the catalogs of the table:

```
SELECT STATISTICS_PROFILE FROM SYSIBM.SYSTABLES WHERE NAME =
      'EMPLOYEE'
```

20. Collect statistics, including distribution statistics, on 30 percent of the rows:

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION
      TABLESAMPLE BERNOULLI(30)
```

21. To control the sample set on which statistics will be collected and to be able to repeatedly use the same sample set, you can do so as follows:

```
RUNSTATS ON TABLE db2user.employee WITH DISTRIBUTION
      TABLESAMPLE BERNOULLI(30) REPEATABLE(4196)
```

Issuing the same statement as above will result in the same set of statistics as long as the data has not changed in the interm.

22. Collect index statistics as well as table statistics on 1.5 percent of the data pages. Only table data pages and not index pages are sampled. In this example 1.5 percent of table data pages are used for the collection of table statistics, while for index statistics all the index pages will be used:

```
RUNSTATS ON TABLE db2user.employee AND INDEXES ALL TABLESAMPLE SYSTEM(1.5)
```

23. Collect statistics for a statistical view, on all columns, without distribution statistics:

```
RUNSTATS ON TABLE salesdb.product_sales_view
```

24. Collect statistics for a statistical view, with distribution statistics on the columns `category`, `type` and `product_key`. Distribution statistics limits are set for the `category` column, while the other columns use a common default:

```
RUNSTATS ON TABLE salesdb.product_sales_view
      WITH DISTRIBUTION ON COLUMNS (category NUM_FREQVALUES 100 NUM_QUANTILES 100,
      type, product_key) DEFAULT NUM_FREQVALUES 50 NUM_QUANTILES 50
```

25. Collect statistics, including distribution statistics, on 10 percent of the rows using row level sampling:
- ```
RUNSTATS ON TABLE db2user.daily_sales
WITH DISTRIBUTION TABLESAMPLE BERNOULLI (10)
```
26. Collect statistics, including distribution statistics, on 2.5 percent of the rows using data page level sampling. Additionally, specify the repeated use of the same sample set. For this command to succeed, the query must be such that the DB2 database system can successfully push data page sampling down to one or more tables. Otherwise, an error (SQL 20288N) is raised.
- ```
RUNSTATS ON TABLE db2user.daily_sales
WITH DISTRIBUTION TABLESAMPLE SYSTEM (2.5)
```
27. Register a statistics profile to collect statistics on the view and on all columns with distribution statistics as specified:
- ```
RUNSTATS ON TABLE salesdb.product_sales_view
WITH DISTRIBUTION DEFAULT NUM_FREQVALUES 50 NUM_QUANTILES 50
SET PROFILE
```
28. Modify the previously registered statistics profile. This command also updates the statistics as specified:
- ```
RUNSTATS ON TABLE salesdb.product_sales_view
WITH DISTRIBUTION DEFAULT NUM_FREQVALUES 25 NUM_QUANTILES 25
UPDATE PROFILE
```

Related concepts:

- “Automatic statistics collection” in *Performance Guide*
- “Collecting statistics on a sample of the table data” in *Performance Guide*
- “Collecting statistics using a statistics profile” in *Performance Guide*

Related tasks:

- “Collecting catalog statistics” in *Performance Guide*

Related reference:

- “ADMIN_COPY_SCHEMA procedure – Copy a specific schema and its objects” in *Administrative SQL Routines and Views*
- “db2Runstats API - Update statistics for tables and indexes” in *Administrative API Reference*
- “RUNSTATS command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

SET CLIENT

Specifies connection settings for the back-end process.

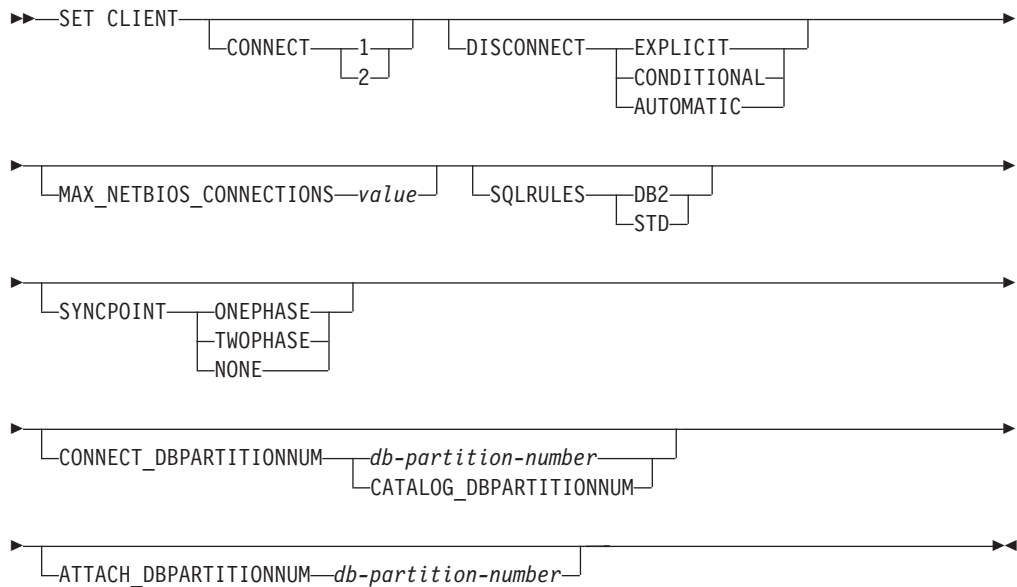
Authorization:

None

Required connection:

None

Command syntax:



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. This command parameter is ignored and is only included here for backward compatibility.

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_DBPARTITIONNUM (partitioned database environment only)**db-partition-number**

Specifies the database partition to which a connect is to be made. A value between zero and 999, inclusive. Overrides the value of the environment variable **DB2NODE**.

CATALOG_DBPARTITIONNUM

Specifying this value permits the client to connect to the catalog database partition of the database without knowing the identity of that database partition in advance.

ATTACH_DBPARTITIONNUM db-partition-number (partitioned database environment only)

Specifies the database partition 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 database partitions 1, 2, and 3 are defined, the client only needs to be able to access one of these database partitions. If only database partition 1 containing databases has been cataloged, and this parameter is set to 3, then the next attach attempt will result in an attachment at database partition 3, after an initial attachment at database partition 1.

Examples:

To set specific values:

SET CLIENT

```
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
```

The connection settings revert to default values after the TERMINATE command 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.

Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword CONNECT_NODE can be substituted for CONNECT_DBPARTITIONNUM.
- The keyword CATALOG_NODE can be substituted for CATALOG_DBPARTITIONNUM.
- The keyword ATTACH_NODE can be substituted for ATTACH_DBPARTITIONNUM.

Related reference:

- “sqlesetc API - Set client connection settings” in *Administrative API Reference*
- “TERMINATE ” on page 744
- “QUERY CLIENT ” on page 611

SET RUNTIME DEGREE

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.

Related tasks:

- “Enabling intra-partition parallelism for queries” in *Administration Guide: Implementation*

Related reference:

- “sqsdeg API - Set the maximum runtime intra-partition parallelism level or degree for SQL statements” in *Administrative API Reference*
- “LIST APPLICATIONS ” on page 520

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.

Authorization:

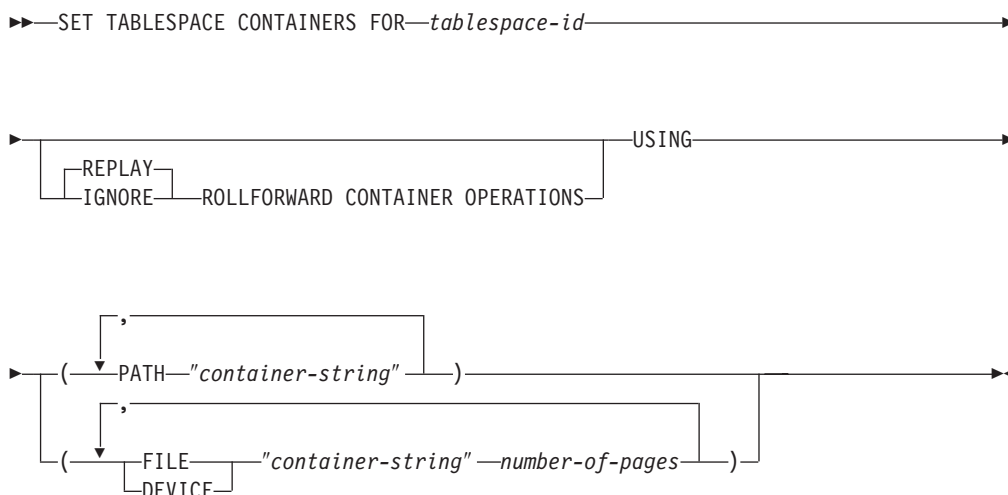
One of the following:

- *sysadm*
- *sysctrl*

Required connection:

Database

Command syntax:



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

SET TABLESPACE CONTAINERS

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 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.

Examples:

See the example in RESTORE DATABASE.

Usage notes:

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.

Related reference:

- "sqlbstsc API - Set table space containers" in *Administrative API Reference*
- "BACKUP DATABASE " on page 349
- "RESTORE DATABASE " on page 675
- "ROLLFORWARD DATABASE " on page 691

SET TAPE POSITION

Sets the positions of tapes for backup and restore operations to streaming tape devices. This command is only supported on Windows operating systems.

Authorization:

One of the following:

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

Required connection:

None.

Command syntax:

```

▶▶ SET TAPE POSITION ON device TO position

```

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 writes a tape mark after every backup image. 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, for example, archive 2 is positioned to be restored.

Related reference:

- “ADMIN_CMD procedure – Run administrative commands” in *Administrative SQL Routines and Views*
- “SET TAPE POSITION command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*
- “INITIALIZE TAPE ” on page 511
- “REWIND TAPE ” on page 690

SET UTIL_IMPACT_PRIORITY

Changes the impact setting for a running utility. Using this command, you can:

- throttle a utility that was invoked in unthrottled mode
- unthrottle a throttled utility (disable throttling)
- reprioritize a throttled utility (useful if running multiple simultaneous throttled utilities)

Scope:

Authorization:

One of the following:

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

Required connection:

Instance.

Command syntax:

```
▶▶—SET UTIL_IMPACT_PRIORITY FOR—utility-id—TO—priority————▶▶
```

Command parameters:

utility-id

ID of the utility whose impact setting will be updated. IDs of running utilities can be obtained with the LIST UTILITIES command.

TO *priority*

Specifies an instance-level limit on the impact associated with running a utility. A value of 100 represents the highest priority and 1 represents the lowest priority. Setting *priority* to 0 will force a throttled utility to continue unthrottled. Setting *priority* to a non-zero value will force an unthrottled utility to continue in throttled mode.

Examples:

The following example unthrottles the utility with ID 2.

```
SET UTIL_IMPACT_PRIORITY FOR 2 TO 0
```

The following example throttles the utility with ID 3 to priority 10. If the priority was 0 before the change then a previously unthrottled utility is now throttled. If the utility was previously throttled (priority had been set to a value greater than zero), then the utility has been reprioritized.

```
SET UTIL_IMPACT_PRIORITY FOR 3 TO 10
```

Usage notes:

Throttling requires having an impact policy defined by setting the *util_impact_lim* configuration parameter.

Related reference:

- “LIST UTILITIES” on page 555
- “util_impact_lim - Instance impact policy configuration parameter” in *Performance Guide*

SET WRITE

The SET WRITE command allows a user to suspend I/O writes or to resume I/O writes for a database. Typical use of this command is for splitting a mirrored database. This type of mirroring is achieved through a disk storage system.

This new state, `SUSPEND_WRITE`, is visible from the Snapshot Monitor. All table spaces must be in a `NORMAL` state for the command to execute successfully. If any one table space is in a state other than `NORMAL`, the command will fail.

Scope:

This command only affects the database partition on which it is executed.

Authorization:

This command only affect the node on which it is executed. The authorization of this command requires the issuer to have one of the following privileges:

- `sysadm`
- `sysctrl`
- `sysmaint`

Required Connection:

Database

Command Syntax:

```

▶▶ SET WRITE {SUSPEND | RESUME} FOR {DATABASE | DB}

```

Command Parameters:

SUSPEND

Suspending I/O writes will put all table spaces into a new state `SUSPEND_WRITE` state. Writes to the logs are also suspended by this command. All database operations, apart from online backup and restore, should function normally while database writes are suspended. However, some operations can wait while attempting to flush dirty pages from the buffer pool or log buffers to the logs. These operations will resume normally once the database writes are resumed.

RESUME

Resuming I/O writes will remove the `SUSPEND_WRITE` state from all of the table spaces and make the table spaces available for update.

Usage Notes:

It is suggested that I/O writes be resumed from the same connection from which they were suspended. Ensuring that this connection is available to resume I/O writes involves not performing any operations from this connection until database writes are resumed. Otherwise, some operations can wait for I/O writes to be resumed if dirty pages must be flushed from the buffer pool or from log buffers to the logs. Furthermore, subsequent connection attempts might hang if they require flushing dirty pages from the buffer pool to disk. Subsequent connections will complete successfully once database I/O resumes. If your connection attempts are

hanging, and it has become impossible to resume I/O from the connection that you used to suspend I/O, then you will have to run the `RESTART DATABASE` command with the `WRITE RESUME` option. When used in this circumstance, the `RESTART DATABASE` command will resume I/O writes without performing crash recovery. The `RESTART DATABASE` command with the `WRITE RESUME` option will only perform crash recovery when you use it after a database crash.

Related concepts:

- “High availability through online split mirror and suspended I/O support” in *Data Recovery and High Availability Guide and Reference*

Related tasks:

- “Using a split mirror as a backup image” in *Data Recovery and High Availability Guide and Reference*
- “Using a split mirror as a standby database” in *Data Recovery and High Availability Guide and Reference*
- “Using a split mirror to clone a database” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “db2SetWriteForDB API - Suspend or resume I/O writes for database” in *Administrative API Reference*

START DATABASE MANAGER

Starts the current database manager instance background processes on a single database partition or on all the database partitions defined in a multi-partitioned database environment.

Scope:

In a multi-partitioned database environment, this command affects all database partitions that are listed in the \$HOME/sqllib/db2nodes.cfg file, unless the *dbpartitionnum* parameter is used.

Authorization:

One of the following:

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

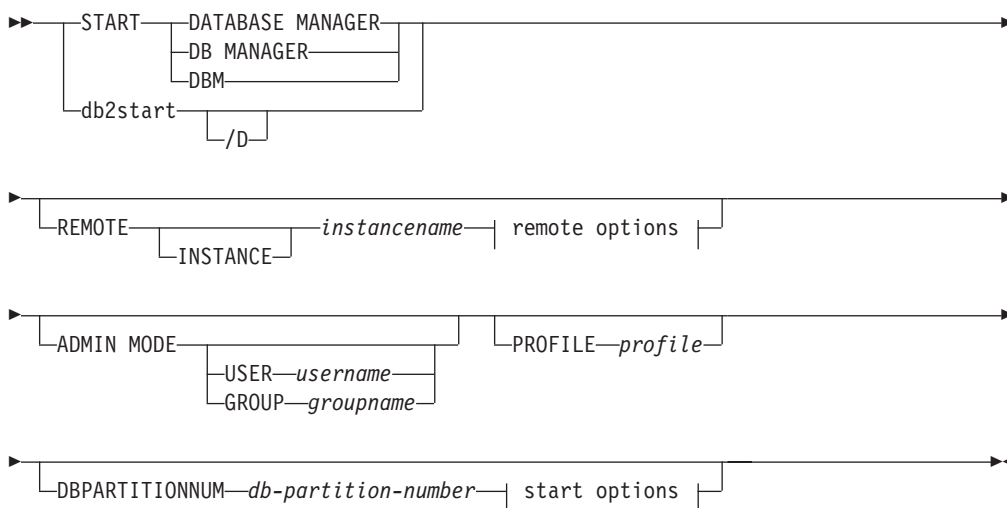
The ADD DBPARTITIONNUM start option requires either *sysadm* or *sysctrl* authority.

You must meet Windows operating system requirements for starting a service. If Extended Security is disabled, you must be a member of the Administrators, Server Operators or Power Users group. If Extended Security is enabled, you must be a member of either the Administrators group or the DB2ADMNS group to start the database.

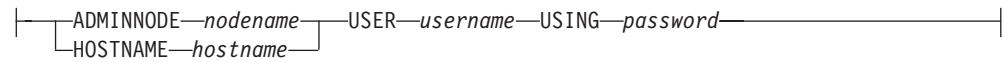
Required connection:

None

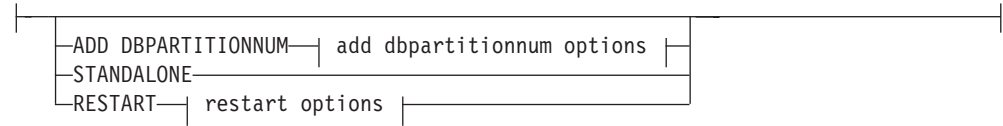
Command syntax:



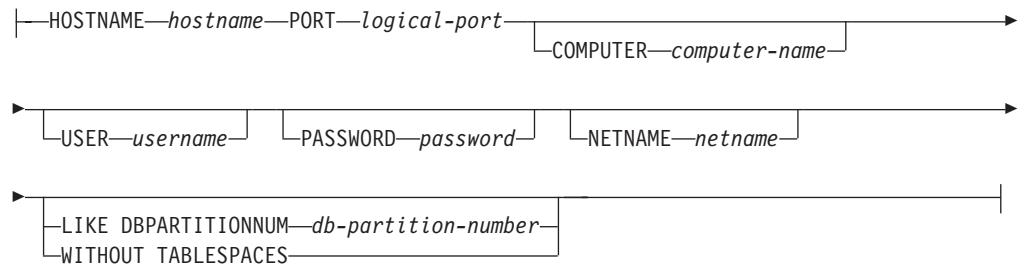
remote options:



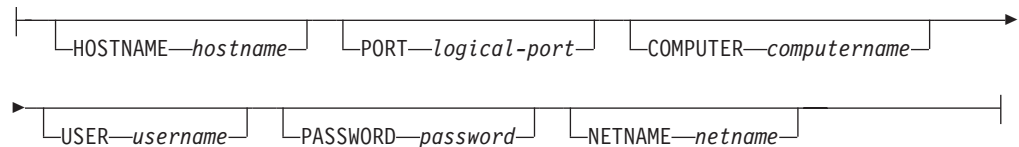
start options:



add dbpartitionnum options:



restart options:



Command parameters:

REMOTE [INSTANCE] instancename

Specifies the name of the remote instance you wish to start.

ADMINNODE nodename

With REMOTE, or REMOTE INSTANCE, specifies the name of the administration node.

HOSTNAME hostname

With REMOTE, or REMOTE INSTANCE, specifies the name of the host node.

USER username

With REMOTE, or REMOTE INSTANCE, specifies the name of the user.

USING password

With REMOTE, or REMOTE INSTANCE, and the USER, specifies the password of the user.

ADMIN MODE

Starts the instance in quiesced mode for administration purposes. This is equivalent to the QUIESCE INSTANCE command except in this case the instance is not already “up”, and therefore there is no need to force the connections off.

START DATABASE MANAGER

USER *username*

With ADMIN MODE, specifies the name of the user.

GROUP *groupname*

With ADMIN MODE, specifies the name of the group.

All of the following parameters are valid in an Enterprise Server Edition (ESE) environment only.

PROFILE *profile*

Specifies the name of the profile file to be executed at each database partition to define the DB2 environment. This file is executed before the database partitions are started. The profile file must reside in the `sql1ib` directory of the instance owner. The environment variables in the profile file are not necessarily all defined in the user session.

DBPARTITIONNUM *db-partition-number*

Specifies the database partition to be started. If no other options are specified, a normal startup is done at this database partition.

Valid values are from 0 to 999 inclusive. If ADD DBPARTITIONNUM is not specified, the value must already exist in the `db2nodes.cfg` file of the instance owner. If no database partition number is specified, all database partitions defined in the configuration file are started.

ADD DBPARTITIONNUM

Specifies that the new database partition 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 database partition utility is executed internally to create all existing databases on the database partition being added. After a database partition is added, the `db2nodes.cfg` file is not updated with the new database partition until a **db2stop** is issued. The database partition is not part of the MPP system until the next **db2start** following the **db2stop**.

When the database partitions are created on the new node, their configuration parameters are set to the default.

HOSTNAME *hostname*

With ADD DBPARTITIONNUM, specifies the host name to be added to the `db2nodes.cfg` file.

PORT *logical-port*

With ADD DBPARTITIONNUM, specifies the logical port to be added to the `db2nodes.cfg` file. Valid values are from 0 to 999.

COMPUTER *computername*

The computer name for the machine on which the new database partition is created. This parameter is mandatory on Windows, but is ignored on other operating systems.

USER *username*

The user name for the account on the new database partition. This parameter is mandatory on Windows, but is ignored on other operating systems.

PASSWORD *password*

The password for the account on the new database partition. This parameter is mandatory on Windows, 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 DBPARTITIONNUM db-partition-number

Specifies that the containers for the system temporary table spaces will be the same as the containers on the specified *db-partition-number* for each database in the instance. The database partition specified must be a database partition that is already in the `db2nodes.cfg` file. For system temporary table spaces that are defined to use automatic storage (in other words, system temporary table spaces that were created with the `MANAGED BY AUTOMATIC STORAGE` clause of the `CREATE TABLESPACE` statement or where no `MANAGED BY CLAUSE` was specified at all), the containers will not necessarily match those from the partition specified. Instead, containers will automatically be assigned by the database manager based on the storage paths that are associated with the database. This may or may not result in the same containers being used on these two partitions.

WITHOUT TABLESPACES

Specifies that containers for the system temporary table spaces are not created for any of the databases. The `ALTER TABLESPACE` statement must be used to add system temporary table space containers to each database before the database can be used. This option is ignored for system temporary table spaces that are defined to use automatic storage (in other words, system temporary table spaces that were created with the `MANAGED BY AUTOMATIC STORAGE` clause of the `CREATE TABLESPACE` statement or where no `MANAGED BY CLAUSE` was specified at all). For these table spaces, there is no way to defer container creation. Containers will automatically be assigned by the database manager based on the storage paths that are associated with the database.

STANDALONE

Specifies that the database partition is to be started in stand-alone mode. FCM does not attempt to establish a connection to any other database partition. This option is used when adding a database partition.

RESTART

Starts the database manager after a failure. Other database partitions are still operating, and this database partition 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 database partitions 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 database partition configuration file.

PORT logical-port

With `RESTART`, specifies the logical port number to be used to override that in the database partition configuration file. If not

START DATABASE MANAGER

specified, this parameter defaults to the *logical-port* value that corresponds to the *num* value in the `db2nodes.cfg` file. Valid values are from 0 to 999.

COMPUTER **computername**

The computer name for the machine on which the new database partition is created. This parameter is mandatory on Windows, but is ignored on other operating systems.

USER **username**

The user name for the account on the new database partition. This parameter is mandatory on Windows, but is ignored on other operating systems.

PASSWORD **password**

The password for the account on the new database partition. This parameter is mandatory on Windows, but is ignored on other operating systems.

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 *db-partition-number* value in the `db2nodes.cfg` file.

Examples:

The following is sample output from **db2start** issued on a three-database partition system with database partitions 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 partitioned database environment, messages are returned on the database partition that issued the START DATABASE MANAGER command.

If no parameters are specified in a partitioned database environment, the database manager is started on all parallel nodes using the parameters specified in the database partition configuration file.

If a START DATABASE MANAGER command is in progress, ensure that the applicable database partitions have started *before* issuing a request to the database.

The `db2cshrc` file is not supported and cannot be used to define the environment.

You can start an instance in a quiesced state. You can do this by using one of the following choices:

```
db2start admin mode
```

or

```
db2start admin mode user username
```

or

```
db2start admin mode group groupname
```

When adding a new database partition, **START DATABASE MANAGER** must determine whether or not each database in the instance is enabled for automatic storage. This is done by communicating with the catalog partition for each database. If automatic storage is enabled then the storage path definitions are retrieved as part of that communication. Likewise, if system temporary table spaces are to be created with the database partitions, **START DATABASE MANAGER** might have to communicate with another database partition server to retrieve the table space definitions for the database partitions that reside on that server. The `start_stop_time` database manager configuration parameter is used to specify the time, in minutes, by which the other database partition server must respond with the automatic storage and table space definitions. If this time is exceeded, the command fails. If this situation occurs, increase the value of `start_stop_time`, and reissue the command.

On UNIX platforms, the **START DATABASE MANAGER** command supports the SIGINT signal. It is issued if CTRL+C is pressed. If this signal occurs, all in-progress startups are interrupted and a message (SQL1044N) is returned from each interrupted database partition to the `$HOME/sql1lib/log/db2start.timestamp.log` error log file. Database partitions that are already started are not affected. If CTRL+C is pressed on a database partition that is starting, **db2stop** must be issued on that database partition before an attempt is made to start it again.

On Windows operating systems, neither the **db2start** command nor the **NET START** command returns warnings if any communication subsystem failed to start. The database manager in a Windows environment is implemented as a service, and does not return an error if the service is started successfully. Be sure to examine the Event Log or the `DB2DIAG.LOG` file for any errors that might have occurred during the running of **db2start**.

Compatibilities:

For compatibility with versions earlier than Version 8:

- The keywords `LIKE NODE` can be substituted for `LIKE DBPARTITIONNUM`.
- The keyword `ADDNODE` can be substituted for `ADD DBPARTITIONNUM`.
- The keyword `NODENUM` can be substituted for `DBPARTITIONNUM`.

Related reference:

- “STOP DATABASE MANAGER ” on page 736
- “ADD DBPARTITIONNUM ” on page 336
- “db2InstanceStart API - Start instance” in *Administrative API Reference*

START HADR

Starts HADR operations for a database.

Authorization:

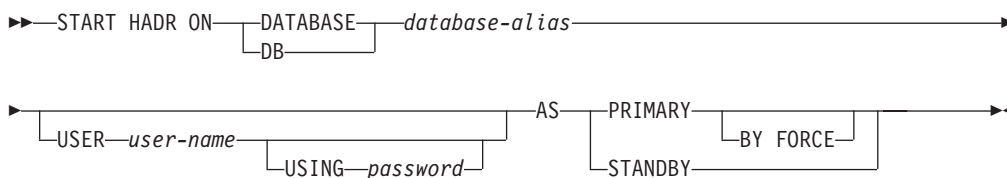
One of the following:

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

Required connection:

Instance. The command establishes a database connection if one does not exist, and closes the database connection when the command completes.

Command syntax:



Command parameters:

DATABASE *database-alias*

Identifies the database on which HADR operations are to start.

USER *user-name*

Identifies the user name under which the HADR operations are to be started.

USING *password*

The password used to authenticate *user-name*.

AS PRIMARY

Specifies that HADR primary operations are to be started on the database.

BY FORCE

Specifies that the HADR primary database will not wait for the standby database to connect to it. After a start BY FORCE, the primary database will still accept valid connections from the standby database whenever the standby later becomes available. When BY FORCE is used, the database will perform crash recovery if necessary, regardless of the value of database configuration parameter AUTORESTART. Other methods of starting a primary database (such as non-forced **START HADR** command, **ACTIVATE DATABASE** command, or client connection) will respect the AUTORESTART setting.

Caution: Use the **START HADR** command with the AS PRIMARY BY FORCE option with caution. If the standby database has been changed to a primary and the original primary database is restarted by issuing the **START HADR** command with the AS PRIMARY BY FORCE option, both

copies of your database will be operating independently as primaries. (This is sometimes referred to as *split brain* or *dual primary*.) In this case, each primary database can accept connections and perform transactions, and neither receives and replays the updates made by the other. As a result, the two copies of the database will become inconsistent with each other.

AS STANDBY

Specifies that HADR standby operations are to be started on the database. The standby database will attempt to connect to the HADR primary database until a connection is successfully established, or until the connection attempt is explicitly rejected by the primary. (The connection might be rejected by the primary database if an HADR configuration parameter is set incorrectly or if the database copies are inconsistent, both conditions for which continuing to retry the connection is not appropriate.)

Usage notes:

The following table shows database behavior in various conditions:

| Database status | Behavior upon START HADR command with the AS PRIMARY option | Behavior upon START HADR command with the AS STANDBY option |
|----------------------------|---|---|
| Inactive standard database | Activated as HADR primary database. | Database starts as an standby database if it is in rollforward-pending mode (which can be the result of a restore or a split mirror) or in rollforward in-progress mode. Otherwise, an error is returned. |
| Active standard database | Database enters HADR primary role. | Error message returned. |
| Inactive primary database | Activated as HADR primary database. | After a failover, this reintegrates the failed primary into the HADR pair as the new standby database. Some restrictions apply. |
| Active primary database | Warning message issued. | Error message returned. |
| Inactive standby database | Error message returned. | Starts the database as the standby database. |
| Active standby database | Error message returned. | Warning message issued. |

When issuing the **START HADR** command, the corresponding error codes might be generated: SQL01767N, SQL01769N, or SQL01770N with a reason code of 98. The reason code indicates that there is no installed license for HADR on the server where the command was issued. To correct the problem, install a valid HADR license using the **db2licm** or install a version of the server that contains a valid HADR license as part of its distribution.

Related tasks:

- “Initializing high availability disaster recovery (HADR)” in *Data Recovery and High Availability Guide and Reference*

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 partitioned database system, this command stops the current database manager instance on a database partition or on all database partitions. When it stops the database manager on all database partitions, it uses the `db2nodes.cfg` configuration file to obtain information about each database partition.

This command can also be used to drop a database partition from the `db2nodes.cfg` file (partitioned database systems only).

This command is not valid on a client.

Scope:

By default, and in a partitioned database environment, this command affects all database partitions that are listed in the `db2nodes.cfg` file.

Authorization:

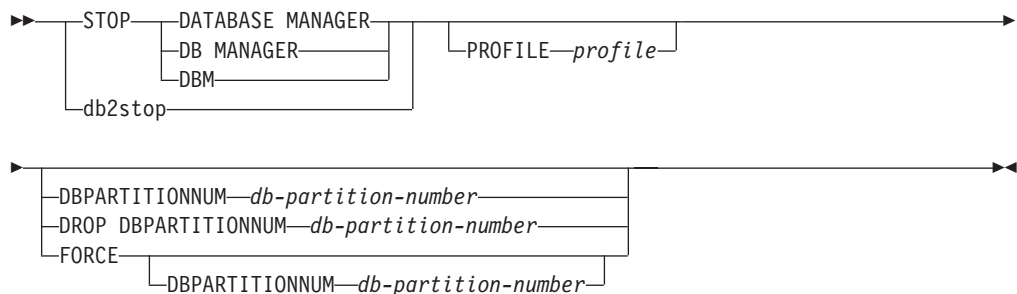
One of the following:

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

Required connection:

None

Command syntax:



Command parameters:

PROFILE profile

partitioned database systems only. Specifies the name of the profile file that was executed at startup to define the DB2 environment for those database partitions that were started. If a profile for the START DATABASE

MANAGER command was specified, the same profile must be specified here. The profile file must reside in the `sqllib` directory of the instance owner.

DBPARTITIONNUM db-partition-number

partitioned database systems only. Specifies the database partition to be stopped.

Valid values are from 0 to 999 inclusive, and must be in the `db2nodes.cfg` file. If no database partition number is specified, all database partitions defined in the configuration file are stopped.

DROP DBPARTITIONNUM db-partition-number

partitioned database systems only. Specifies the database partition to be dropped from the `db2nodes.cfg` file.

Before using this parameter, run the `DROP DBPARTITIONNUM VERIFY` command to ensure that there is no user data on this database partition.

When this option is specified, all database partitions in the `db2nodes.cfg` file are stopped.

FORCE

Specifies to use `FORCE APPLICATION ALL` when stopping the database manager at each database partition.

DBPARTITIONNUM db-partition-number

partitioned database systems only. Specifies the database partition to be stopped after all applications on that database partition have been forced to stop. If the `FORCE` option is used without this parameter, all applications on all database partitions are forced before all the database partitions are stopped.

Examples:

The following is sample output from `db2stop` issued on a three-partition system with database partitions 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 the `FORCE APPLICATION` command to disconnect all users first, or reissue the `STOP DATABASE MANAGER` command with the `FORCE` option.

STOP DATABASE MANAGER

The following information applies to partitioned database environments only:

- If no parameters are specified, the database manager is stopped on each database partition listed in the configuration file. The administration notification log might contain messages to indicate that other database partitions are shutting down.
- Any database partitions added to the partitioned database system since the previous STOP DATABASE MANAGER command was issued will be updated in the db2nodes.cfg file.
- On UNIX platforms, if the value specified for the *start_stop_time* database manager configuration parameter is reached, all in-progress stops are interrupted, and message SQL6037N is returned from each interrupted database partition to the \$HOME/sqllib/log/db2stop.*timestamp*.log error log file. Database partitions 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.

Related reference:

- “FORCE APPLICATION ” on page 439
- “START DATABASE MANAGER ” on page 728
- “DEACTIVATE DATABASE ” on page 411
- “DROP DBPARTITIONNUM VERIFY ” on page 428
- “db2InstanceStop API - Stop instance” in *Administrative API Reference*

STOP HADR

Stops HADR operations for a database.

Authorization:

One of the following:

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

Required connection:

Instance. The command establishes a database connection if one does not exist, and closes the database connection when the command completes.

Command syntax:

```

▶▶ STOP HADR ON DATABASE database-alias
DB

```

```

USER user-name
USING password

```

Command parameters:

DATABASE *database-alias*

Identifies the database on which HADR operations are to stop.

USER *user-name*

Identifies the user name under which the HADR operations are to be stopped.

USING *password*

The password used to authenticate *user-name*.

Usage notes:

The following table shows database behavior in various conditions:

| Database status | Behavior upon STOP HADR command |
|----------------------------|--|
| Inactive standard database | Error message returned. |
| Active standard database | Error message returned. |
| Inactive primary database | Database role changes to standard. Database configuration parameter <i>hadr_db_role</i> is updated to STANDARD. Database remains offline. At the next restart, enters standard role. |

STOP HADR

| Database status | Behavior upon STOP HADR command |
|---------------------------|---|
| Active primary database | Stops shipping logs to the HADR standby database and shuts down all HADR EDUs on the HADR primary database. Database role changes to standard and database remains online. Database remains in standard role until an explicit START HADR command with the AS PRIMARY option is issued. Open sessions and transactions are not affected by the STOP HADR command. You can repeatedly issue STOP HADR and START HADR commands while the database remains online. These commands take effect dynamically. |
| Inactive standby database | Database role changes to standard. Database configuration parameter <i>hadr_db_role</i> is updated to STANDARD. Database remains offline. Database is put into rollforward pending mode. |
| Active standby database | Error message returned: Deactivate the standby database before attempting to convert it to a standard database. |

When issuing the **STOP HADR** command, the corresponding error codes might be generated: SQL01767N, SQL01769N, or SQL01770N with a reason code of 98. The reason code indicates that there is no installed license for HADR on the server where the command was issued. To correct the problem, install a valid HADR license using the **db2licm** or install a version of the server that contains a valid HADR license as part of its distribution.

Related tasks:

- “Stopping high availability disaster recovery (HADR)” in *Data Recovery and High Availability Guide and Reference*

TAKEOVER HADR

Instructs an HADR standby database to take over as the new HADR primary database for the HADR pair.

Authorization:

One of the following:

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

Required connection:

Instance. The command establishes a database connection if one does not exist, and closes the database connection when the command completes.

Command syntax:

```

▶▶ TAKEOVER HADR ON DATABASE database-alias
DB
USER user-name
USING password
BY FORCE
  
```

Command parameters:

DATABASE *database-alias*

Identifies the current HADR standby database that should take over as the HADR primary database.

USER *user-name*

Identifies the user name under which the takeover operation is to be started.

USING *password*

The password used to authenticate *user-name*.

BY FORCE

Specifies that the database will not wait for confirmation that the original HADR primary database has been shut down. This option is required if the HADR pair is not in peer state.

Usage notes:

The following table shows the behavior of the **TAKEOVER HADR** command when issued on an active standby for each possible state and option combination. An error message is returned if this command is issued on an inactive standby database.

| Standby state | BY FORCE option used | Takeover behavior |
|---------------------------------|----------------------|------------------------|
| Local catchup or remote catchup | No | Error message returned |

TAKEOVER HADR

| Standby state | BY FORCE option used | Takeover behavior |
|---------------------------------|----------------------|---|
| Local catchup or remote catchup | Yes | Error message returned |
| Peer | No | <p>Primary database and standby database switch roles.</p> <p>If no failure is encountered during takeover, there will be no data loss. However, if failures are encountered during takeover, data loss might occur and the roles of the primary and standby might or might not have been changed. The following is a guideline for handling failures during a takeover in which the primary and standby switch roles:</p> <ol style="list-style-type: none"> 1. If a failure occurs during a takeover operation, the roles of the HADR databases might or might not have been changed. If possible, make sure both databases are online. Check the HADR role of the available database or databases using the Snapshot Monitor, or by checking the value of the database configuration parameter <code>hadr_db_role</code>. 2. If the intended new primary is still in standby role, and takeover is still desired, re-issue the TAKEOVER HADR command (see the next guideline regarding the BY FORCE option). 3. It is possible to end up with both databases in standby role. In that case, the TAKEOVER HADR command with the BY FORCE option can be issued at whichever node should now become the primary. The BY FORCE option is required in this case because the two standbys cannot establish the usual HADR primary-standby connection. |
| Peer | Yes | <p>The standby notifies the primary to shut itself (the primary) down. The standby stops receiving logs from the primary, finishes replaying the logs it has already received, and then becomes a primary. The standby does <i>not</i> wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down. Because of this, if the primary is processing transactions at the time of the takeover, it is unlikely that it will be able to be later restarted as a standby. It is recommended that you shut down the primary database first before issuing a TAKEOVER HADR command with the BY FORCE option.</p> |
| Remote catchup pending | No | Error message returned. |
| Remote catchup pending | Yes | The standby becomes a primary. |

When issuing the **TAKEOVER HADR** command, the corresponding error codes might be generated: SQL01767N, SQL01769N, or SQL01770N with a reason code of 98. The reason code indicates that there is no installed license for HADR on the server where the command was issued. To correct the problem, install a valid HADR license using the **db2licm** or install a version of the server that contains a valid HADR license as part of its distribution.

Related tasks:

- “Switching database roles in high availability disaster recovery (HADR)” in *Data Recovery and High Availability Guide and Reference*

TERMINATE

TERMINATE

Explicitly terminates the command line processor's back-end process.

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 prior to executing the **db2stop** command. 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 database partition number within an MPP multiple logical node configuration.

Related reference:

- "db2stop - Stop DB2 " on page 272

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 local database directory can be uncataloged. Entries in the system database directory can be deleted using the **DROP DATABASE** command.

To recatalog the database on the instance, use the **UNCATALOG DATABASE** and **CATALOG DATABASE** commands. To list the databases that are cataloged on a node, use the **LIST DATABASE DIRECTORY** command.

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.

If directory caching is enabled, database, node, and DCS directory files are cached in memory. See the information for the configuration parameter *dir_cache* in the **GET DATABASE MANAGER CONFIGURATION** command. An application's directory cache is created during its first directory lookup. Because the cache is only refreshed when the application modifies any of the directory files, directory changes made by other applications might not be effective until the application has restarted.

To refresh the CLP's directory cache, use the **TERMINATE** command. 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.

Related reference:

- "sqlleuncd API - Uncatalog a database from the system database directory" in *Administrative API Reference*
- "CATALOG DATABASE " on page 372
- "DROP DATABASE " on page 426

UNCATALOG DATABASE

- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "LIST DATABASE DIRECTORY " on page 523
- "TERMINATE " on page 744

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 and can be uncataloged using the **UNCATALOG DATABASE** command.

To recatalog a database in the DCS directory, use the **UNCATALOG DCS DATABASE** and **CATALOG DCS DATABASE** commands. To list the DCS databases that are cataloged on a node, use the **LIST DCS DIRECTORY** command.

If directory caching is enabled, database, node, and DCS directory files are cached in memory. See the information provided for the configuration parameter *dir_cache* in the output of the **GET DATABASE MANAGER CONFIGURATION** command. 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 might not be effective until the application has restarted.

To refresh the CLP's directory cache, use the **TERMINATE** command. 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.

Related reference:

- "sqlgdel API - Uncatalog a database from the database connection services (DCS) directory" in *Administrative API Reference*
- "CATALOG DCS DATABASE " on page 375
- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "TERMINATE " on page 744
- "UNCATALOG DATABASE " on page 745

UNCATALOG DCS DATABASE

- "LIST DCS DIRECTORY " on page 531

UNCATALOG LDAP DATABASE

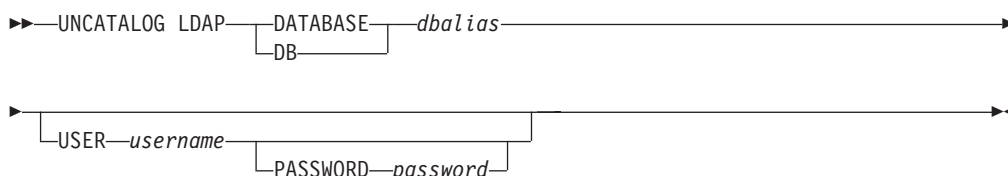
Used to deregister the database from Lightweight Directory Access Protocol (LDAP).

Authorization:

None

Required connection:

None

Command syntax:**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 might, 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.

Related concepts:

- "Lightweight Directory Access Protocol (LDAP) overview" in *Administration Guide: Implementation*

Related tasks:

- "Deregistering the database from the LDAP directory" in *Administration Guide: Implementation*

UNCATALOG LDAP DATABASE

Related reference:

- “db2LdapUncatalogDatabase API - Deregister database from LDAP server” in *Administrative API Reference*
- “CATALOG LDAP DATABASE ” on page 377
- “CATALOG LDAP NODE ” on page 381
- “UNCATALOG LDAP NODE ” on page 751

UNCATALOG LDAP NODE

Uncatalogs a node entry in Lightweight Directory Access Protocol (LDAP).

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.

Related concepts:

- "Lightweight Directory Access Protocol (LDAP) overview" in *Administration Guide: Implementation*

Related reference:

- "db2LdapUncatalogNode API - Delete alias for node name from LDAP server" in *Administrative API Reference*
- "CATALOG LDAP DATABASE " on page 377
- "UNCATALOG LDAP DATABASE " on page 749
- "CATALOG LDAP NODE " on page 381

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.

If directory caching is enabled, 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 might not be effective until the application has restarted.

To refresh the CLP's directory cache, use TERMINATE. 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.

Related reference:

- "sqluncn API - Uncatalog an entry from the node directory" in *Administrative API Reference*
- "CATALOG LOCAL NODE " on page 382
- "CATALOG NAMED PIPE NODE " on page 384
- "CATALOG TCPIP/TCPIP4/TCPIP6 NODE " on page 387
- "GET DATABASE MANAGER CONFIGURATION " on page 463
- "TERMINATE " on page 744

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. That name is used to access the database through ODBC APIs. On Windows, 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 only.

Authorization:

None

Required connection:

None

Command syntax:

```

▶▶—UNCATALOG—

USER
SYSTEM

—ODBC DATA SOURCE—data-source-name—▶▶
  
```

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.

Related reference:

- “CATALOG ODBC DATA SOURCE ” on page 386
- “LIST ODBC DATA SOURCES ” on page 544

UNQUIESCE

Restores user access to instances or databases which have been quiesced for maintenance or other reasons. UNQUIESCE restores user access without necessitating a shutdown and database restart.

Unless specifically designated, no user except those with *sysadm*, *sysmaint*, or *sysctrl* has access to a database while it is quiesced. Therefore an UNQUIESCE is required to restore general access to a quiesced database.

Scope:

UNQUIESCE DB restores user access to all objects in the quiesced database.

UNQUIESCE INSTANCE *instance-name* restores user access to the instance and the databases in the instance *instance-name*.

To stop the instance and unquiesce it and all its databases, issue the `db2stop` command. Stopping and restarting DB2 will unquiesce all instances and databases.

Authorization:

One of the following:

For database level unquiesce:

- *sysadm*
- *dbadm*

For instance level unquiesce:

- *sysadm*
- *sysctrl*

Command syntax:

```

>> UNQUIESCE — DB —————>>
                |
                |—— INSTANCE — instance-name —|
  
```

Required connection:

Database

(Database connection is not required for an instance unquiesce.)

Command parameters:

DB Unquiesce the database. User access will be restored to all objects in the database.

INSTANCE *instance-name*
Access is restored to the instance *instance-name* and the databases in the instance.

Examples:

Unquiescing a Database

```
db2 unquiesce db
```

This command will unquiesce the database that had previously been quiesced.

Related reference:

- “QUIESCE ” on page 612
- “db2DatabaseUnquiesce API - Unquiesce database” in *Administrative API Reference*
- “db2InstanceUnquiesce API - Unquiesce instance” in *Administrative API Reference*
- “UNQUIESCE DATABASE command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE ADMIN CONFIGURATION

Modifies specified entries in the DB2 Administration Server (DAS) configuration file. The DAS is a special administrative tool that enables remote administration of DB2 servers.

When you install the DAS, a blank copy of the configuration file is stored on each physical database partition. You must create entries in each copy. You can specify the following DAS configuration parameters to be used the next time you start the DAS:

- Name of the DB2 Server System - `db2system`
- DAS Administration Authority Group Name - `dasadm_group`
- Scheduler Mode - `sched_enable`
- Tools Catalog Database Instance - `toolscat_inst`
- Tools Catalog Database - `toolscat_db`
- Tools Catalog Database Schema - `toolscat_schema`
- Execute Expired Tasks - `exec_exp_task`
- Scheduler User ID - `sched_userid`
- Authentication Type DAS - `authentication`

The following DAS configuration parameters can be specified originally and then later changed while the DAS is online:

- DAS Discovery Mode - `discover`
- SMTP Server - `smtp_server`
- Java Development Kit Installation Path DAS - `jdk_path`
- Location of Contact List - `contact_host`
- DAS Code Page - `das_codepage`
- DAS Territory - `das_territory`

For more information about these parameters, see individual parameter descriptions.

Scope:

Issue this command from each administration node to specify or change parameter settings for that node.

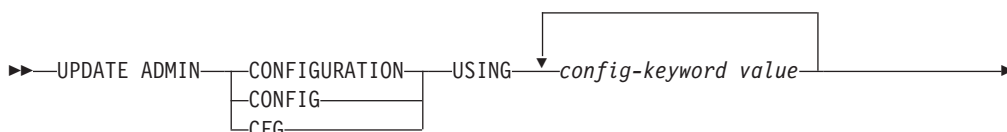
Authorization:

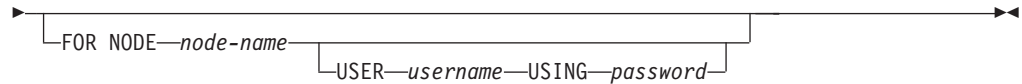
dasadm

Required connection:

Node. To update the DAS configuration for a remote system, use the FOR NODE option with the administrator node name.

Command syntax:





Command parameters:

USING config-keyword value

Specifies the admin configuration parameter to be updated.

FOR NODE

Enter the name of an administration node to update the DAS configuration parameters there.

USER username USING password

If connection to the administration node requires user name and password authorization, enter this information.

Usage notes:

To view or print a list of the DAS configuration parameters, use `GET ADMIN CONFIGURATION`. To reset the DAS configuration parameters to the recommended DAS defaults, use `RESET ADMIN CONFIGURATION`.

When configuration parameters take effect depends on whether you change a standard configuration parameter or one of the parameters that can be reset online. Standard configuration parameter values are reset when you execute the **db2admin** command.

If an error occurs, the DAS configuration file is not changed.

In order to update the DAS configuration using `UPDATE ADMIN CONFIGURATION`, you must use the command line processor from an instance that is at the same installed level as the DAS.

The DAS configuration file cannot be updated if the checksum is invalid. This might occur if you change the DAS configuration file manually, without using the appropriate command. If this happens, you must drop and re-create the DAS to reset its configuration file.

Related reference:

- “`GET ADMIN CONFIGURATION` ” on page 441
- “`RESET ADMIN CONFIGURATION` ” on page 663
- “Configuration parameters summary” in *Performance Guide*

UPDATE ALERT CONFIGURATION

Updates the alert configuration settings for health indicators.

Authorization:

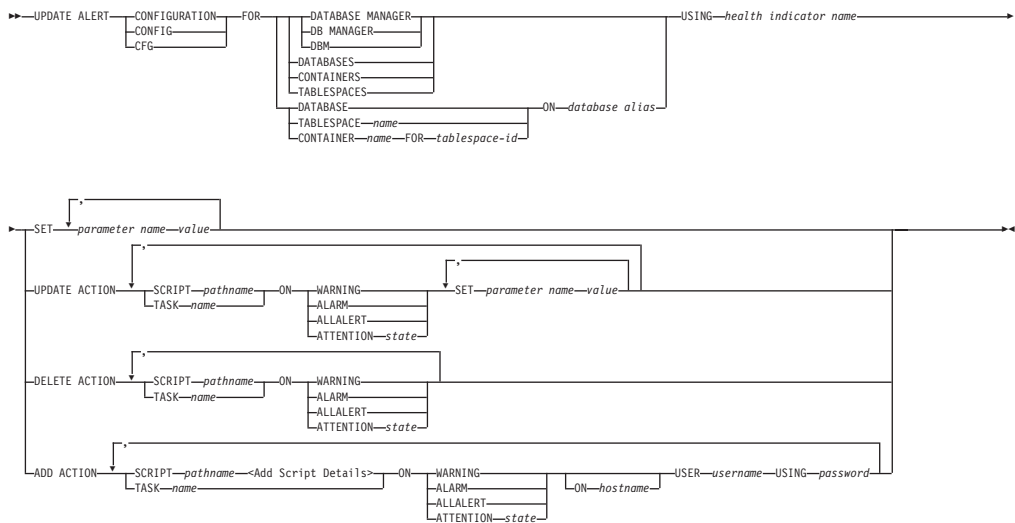
One of the following:

- sysadm
- sysmaint
- sysctrl

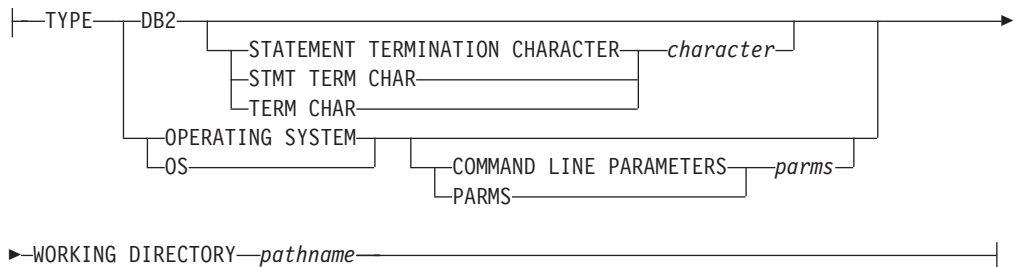
Required Connection:

Instance. An explicit attachment is not required.

Command Syntax:



Add Script Details:



Command Parameters:

DATABASE MANAGER

Updates alert settings for the database manager.

DATABASES

Updates alert settings for all databases managed by the database manager.

These are the settings that apply to all databases that do not have custom settings. Custom settings are defined using the "DATABASE ON database alias" clause.

CONTAINERS

Updates alert settings for all table space containers managed by the database manager. These are the settings that apply to all table space containers that do not have custom settings. Custom settings are defined using the "CONTAINER name ON database alias" clause.

TABLESPACES

Updates alert settings for all table spaces managed by the database manager. These are the settings that apply to all table spaces that do not have custom settings. Custom settings are defined using the "TABLESPACE name ON database alias" clause.

DATABASE ON *database alias*

Updates the alert settings for the database specified using the "ON database alias" clause. If this database has custom settings, then they override the settings for all databases for the instance, which is specified using the DATABASES parameter.

CONTAINER *name* FOR *tablespace-id* ON *database alias*

Updates the alert settings for the table space container called *name*, for the table space specified using the "FOR tablespace-id" clause, on the database specified using the "ON database alias" clause. If this table space container has custom settings, then they override the settings for all table space containers for the database, which is specified using the CONTAINERS parameter.

TABLESPACE *name* ON *database alias*

Updates the alert settings for the table space called *name*, on the database specified using the "ON database alias" clause. If this table space has custom settings, then they override the settings for all table spaces for the database, which is specified using the TABLESPACES parameter.

USING *health indicator name*

Specifies the set of health indicators for which alert configuration will be updated. Health indicator names consist of a two-letter object identifier followed by a name which describes what the indicator measures. For example:

```
db.sort_privmem_util
```

SET *parameter-name value*

Updates the alert configuration element, *parameter-name*, of the health indicator to the specified *value*. *parameter-name* must be one of the following:

- ALARM: the *value* is a health indicator unit.
- WARNING: the *value* is a health indicator unit.
- SENSITIVITY: the *value* is in seconds.
- ACTIONSENABLED: the *value* can be either YES or NO.
- THRESHOLDSCHECKED: the *value* can be either YES or NO.

UPDATE ACTION SCRIPT *pathname* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Specifies that the script attributes of the predefined script with absolute *pathname* will be updated according to the following clause:

UPDATE ALERT CONFIGURATION

SET *parameter-name value*

Updates the script attribute, *parameter-name*, to the specified value. *parameter-name* must be one of the following:

- SCRIPTTYPE
OS or DB2 are the valid types.
- WORKINGDIR
- TERMCHAR
- CMDLINEPARMS

The command line parameters that you specify for the operating system script will precede the default supplied parameters. The parameters that are sent to the operating system script are:

- List of user supplied parameters
- Health indicator short name
- Fully qualified object name
- Health indicator value
- Alert state
- USERID
- PASSWORD
- SYSTEM

UPDATE ACTION TASK *name* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Specifies that the task attributes of the task with name *name* will be updated according to the following clause:

SET *parameter-name value*

Updates the task attribute, *parameter-name*, to the specified value. *parameter-name* must be one of the following:

- USERID
- PASSWORD
- SYSTEM

DELETE ACTION SCRIPT *pathname* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Removes the action script with absolute pathname *pathname* from the list of alert action scripts.

DELETE ACTION TASK *name* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Removes the action task called *name* from the list of alert action tasks.

ADD ACTION SCRIPT *pathname* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Specifies that a new action script with absolute pathname *pathname* is to be added, the attributes of which are given by the following:

TYPE An action script must be either a DB2 Command script or an operating system script:

- DB2
- OPERATING SYSTEM

If it is a DB2 Command script, then the following clause allows one to optionally specify the character, *character*, that is used in the script to terminate statements:

STATEMENT TERMINATION CHARACTER ;

UPDATE ALERT CONFIGURATION

If it is an operating system script, then the following clause allows one to optionally specify the command-line parameters, *parms*, that would be passed to the script upon invocation: **COMMAND LINE PARAMETERS** *parms*

WORKING DIRECTORY *pathname*

Specifies the absolute pathname, *pathname*, of the directory in which the script will be executed.

USER *username* **USING** *password*

Specifies the user account, *username*, and associated password, *password*, under which the script will be executed.

ADD ACTION TASK *name* **ON** [WARNING | ALARM | ALLALERT | ATTENTION*state*]

Specifies that a new task, called *name*, is to be added to be run **ON** the specified condition

ON [WARNING | ALARM | ATTENTION *state*]

Specifies the condition on which the action will run. For threshold-based HIs, this is WARNING or ALARM. For state-based HIs, this will be a numeric state as documented in a table to be provided for each state-based HI (for example, table space states).

Related tasks:

- “Configuring health indicators using a client application” in *System Monitor Guide and Reference*

Related reference:

- “ADMIN_CMD procedure – Run administrative commands” in *Administrative SQL Routines and Views*
- “db2UpdateAlertCfg API - Update the alert configuration settings for health indicators” in *Administrative API Reference*
- “UPDATE ALERT CONFIGURATION command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE ALTERNATE SERVER FOR DATABASE

Updates the alternate server for a database alias in the system database directory.

Scope:

This command only affects the database partition on which it is executed.

Authorization:

One of the following:

- *sysadm*
- *sysctrl*

Required connection:

None.

Command syntax:

```

▶▶ UPDATE ALTERNATE SERVER FOR DATABASE database-alias USING
    └──┬──┘
    DB
▶ HOSTNAME hostname PORT port-number
    
```

Command parameters:

DATABASE *database-alias*

Specifies the alias of the database where the alternate server is to be updated.

HOSTNAME *hostname*

Specifies a fully qualified host name or the IP address of the node where the alternate server for the database resides.

PORT *port-number*

Specifies the port number of the alternate server of the database manager instance.

Examples:

The following example updates the alternate server for the SAMPLE database using host name montero and port 20396:

```
db2 update alternate server for database sample using hostname montero port 20396
```

The following two examples reset the alternate server for the SAMPLE database:

```
db2 update alternate server for database sample using hostname NULL port NULL
```

or

```
db2 update alternate server for database sample using hostname "" port NULL
```

Usage notes:

- This command is only applied to the system database directory.
- This command should only be used on a server instance. If it is issued on a client instance, it is ignored and message SQL1889W is returned.

UPDATE ALTERNATE SERVER FOR DATABASE

- If Lightweight Directory Access Protocol (LDAP) support is enabled on the machine on which the command is issued, the alternate server for the database will be automatically registered in the LDAP directory.

Related reference:

- “db2UpdateAlternateServerForDB API - Update the alternate server for a database alias in the system database directory” in *Administrative API Reference*
- “CREATE DATABASE ” on page 395

UPDATE ALTERNATE SERVER FOR LDAP DATABASE

Updates the alternate server for a database in Lightweight Directory Access Protocol (LDAP).

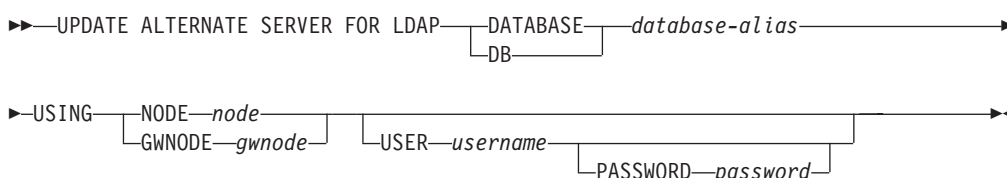
Authorization:

Read/write access to the LDAP server.

Required connection:

None.

Command syntax:



Command parameters:

DATABASE *database-alias*

Specifies the alias of the database to be updated.

NODE *node*

Specifies the node name where the alternate server for the database resides.

GWNODE *gwnode*

Specifies the node name where the alternate gateway for the database resides.

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.

If the user's LDAP DN and password have been specified using **db2ldcfg**, the user name and password do not have to be specified here.

PASSWORD *password*

Account password.

If the user's LDAP DN and password have been specified using **db2ldcfg**, the user name and password do not have to be specified here.

Related tasks:

- "Rerouting LDAP clients to another server" in *Administration Guide: Implementation*

Related reference:

- "CATALOG LDAP DATABASE " on page 377
- "db2ldcfg - Configure LDAP environment " on page 140

UPDATE ALTERNATE SERVER FOR LDAP DATABASE

- “db2LdapUpdateAlternateServerForDB API - Update the alternate server for the database on the LDAP server” in *Administrative API Reference*

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.

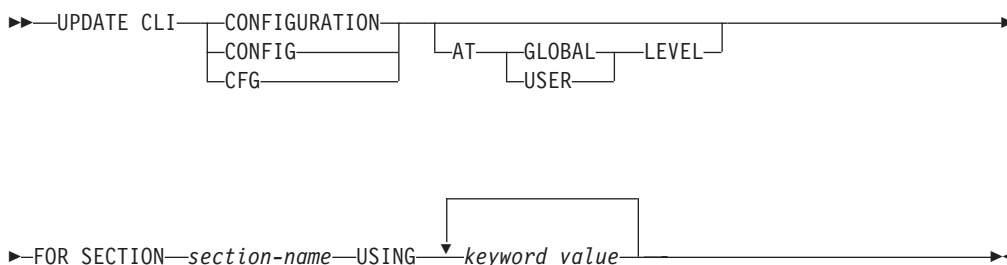
Authorization:

None

Required connection:

None

Command syntax:



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 GLOBAL LEVEL

Specifies that the CLI configuration parameter is to be updated at the global level. This parameter is only applicable when LDAP support is enabled.

AT USER LEVEL

Specifies that the CLI configuration parameter is to be updated at the user level. If LDAP support is enabled, this setting will be consistent when logging on to different machines with the same LDAP user ID. If LDAP support is disabled, this setting will be consistent only when logging on to the same machine with the same operating system user ID.

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 USER LEVEL 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.

In an LDAP environment, users can configure a set of default CLI settings for a database catalogued in the LDAP directory. When an LDAP cataloged database is added as a DSN (Data Source Name), either by using the CA (Configuration Assistant) or the ODBC configuration utility, any default CLI settings, if they exist in the LDAP directory, will be configured for that DSN on the local machine. The AT GLOBAL LEVEL clause must be specified to configure a CLI parameter as a default setting.

Related reference:

- “GET CLI CONFIGURATION ” on page 451
- “REFRESH LDAP ” on page 636

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.

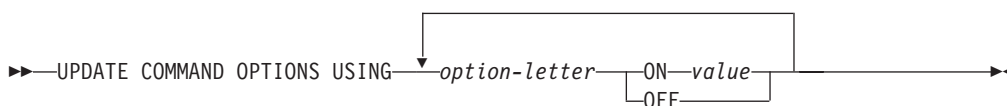
Authorization:

None

Required connection:

None

Command syntax:



Command parameters:

USING option-letter

The following option-letters can be set:

- a** Display SQLCA
- c** Auto-commit SQL statements
- d** Display the XML declarations of XML data
- e** Display SQLCODE/SQLSTATE
- i** Display XQuery results with proper indentation
- l** Log commands in a history file
- m** Display the number of rows affected by INSERT, DELETE, UPDATE or MERGE statements.
- n** Remove new line character
- o** Display to standard output
- p** Display DB2 interactive prompt
- q** Preserve whitespace and line feeds in strings delimited with single or double quotes
- 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

UPDATE COMMAND OPTIONS

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 the **LIST COMMAND OPTIONS** command.

Related reference:

- “LIST COMMAND OPTIONS ” on page 522

UPDATE CONTACT

Updates the attributes of a contact that is defined on the local system. A contact is a user to whom the Scheduler and Health Monitor send messages. To create a contact, use the **ADD CONTACT** command. The setting of the Database Administration Server (DAS) *contact_host* configuration parameter determines whether the list is local or global.

Authorization:

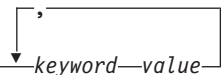
None.

Required connection:

None. Local execution only: this command cannot be used with a remote connection.

Command syntax:

```
►► UPDATE CONTACT name USING keyword value ◄◄
```



Command parameters:

CONTACT **name**

The name of the contact that will be updated.

USING **keyword value**

Specifies the contact parameter to be updated (*keyword*) and the value to which it will be set (*value*). The valid set of keywords is:

ADDRESS

The email address that is used by the SMTP server to send the notification.

TYPE Whether the address is for an email address or a pager.

MAXPAGELEN

The maximum number of characters that the pager can accept.

DESCRIPTION

A textual description of the contact. This has a maximum length of 128 characters.

Related reference:

- “db2UpdateContact API - Update the attributes of a contact” in *Administrative API Reference*
- “UPDATE CONTACT command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE CONTACTGROUP

Updates the attributes of a contact group that is defined on the local system. A contact group is a list of users who should be notified by the Scheduler and the Health Monitor. The setting of the Database Administration Server (DAS) *contact_host* configuration parameter determines whether the list is local or global.

Authorization:

None

Required Connection:

None

Command Syntax:



Command Parameters:

CONTACTGROUP *name*

Name of the contact group which will be updated.

ADD CONTACT *name*

Specifies the name of the new contact to be added to the group. A contact can be defined with the ADD CONTACT command after it has been added to a group.

DROP CONTACT *name*

Specifies the name of a contact in the group that will be dropped from the group.

ADD GROUP *name*

Specifies the name of the new contact group to be added to the group.

DROP GROUP *name*

Specifies the name of a contact group that will be dropped from the group.

DESCRIPTION *new description*

Optional. A new textual description for the contact group.

Related reference:

- “db2UpdateContactGroup API - Update the attributes of a contact group” in *Administrative API Reference*
- “UPDATE CONTACTGROUP command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE DATABASE CONFIGURATION

Modifies individual entries in a specific database configuration file.

A database configuration file resides on every database partition on which the database has been created.

Scope:

This command only affects the database partition on which it is executed.

Authorization:

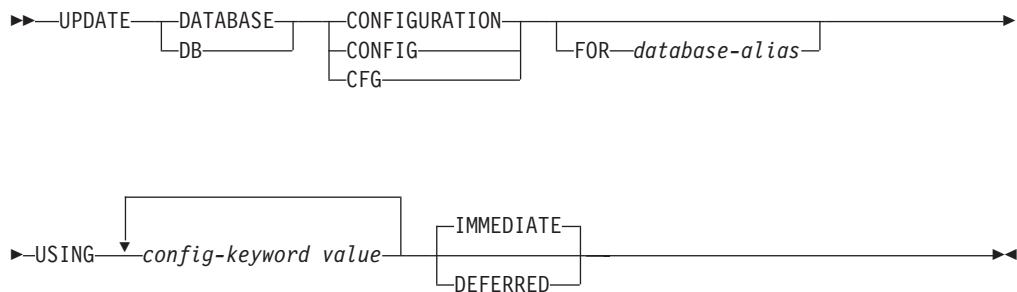
One of the following:

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

Required connection:

Instance. An explicit attachment is not required, but a database connection is recommended when the database is active. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command. To change a parameter online, you must be connected to the database.

Command syntax:



Command parameters:

DEFERRED

Make the changes only in the configuration file, so that the changes take effect the next time you reactivate the database.

FOR database-alias

Specifies the alias of the database whose configuration is to be updated. Specifying the database alias is not required when a database connection has already been established. You can update the configuration file for another database residing under the same database instance. For example, if you are connected only to database db11, and issue update db config for alias db22 using immediate:

- If there is no active connection on db22, the update will be successful because only the configuration file needs to be updated. A new connection (which will activate the database) will see the new change in memory.

UPDATE DATABASE CONFIGURATION

- If there are active connections on db22 from other applications, the update will work on disk but not in memory. You will receive a warning saying that the database needs to be restarted.

IMMEDIATE

Make the changes immediately, while the database is running.

IMMEDIATE is the default action, but it requires a database connection to be effective.

USING *config-keyword value*

config-keyword specifies the database configuration parameter to be updated. *value* specifies the value to be assigned to the parameter.

Usage notes:

To view or print a list of the database configuration parameters, use the GET DATABASE CONFIGURATION command.

To reset all the database configuration parameters to the recommended defaults, use the RESET DATABASE CONFIGURATION command.

To change a database configuration parameter, use the UPDATE DATABASE CONFIGURATION command. For example, to change the logging mode to “archival logging” on a single-partition database environment containing a database called ZELLMART, use:

```
db2 update db cfg for zellmart using logretain recovery
```

To check that the *logretain* configuration parameter has changed, use:

```
db2 get db cfg for zellmart
```

When changing configuration parameters in a multiple-partitioned database environment, the **db2_all** command should be used. Using the **db2_all** command results in the update being issued against all database partitions.

For example, to change the logging mode to “archival logging” in a multiple-partitioned database environment containing a database called “zellmart”, use:

```
db2_all ";db2 update db cfg for zellmart using logretain recovery"
```

To check that the *logretain* configuration parameter has changed on all database partitions, use:

```
db2_all ";db2 get db cfg for zellmart"
```

Optionally, you can leverage the SYSIBMADM.DBCFG view to get data from all partitions without having to use db2_all.

If you are working on a UNIX operating system, and you have the “grep” command, you can use the following command to view only the *logretain* values:

```
db2_all ";db2 get db cfg for zellmart | grep -i logretain"
```

For more information about DB2 configuration parameters and the values available for each type of database node, see the individual configuration parameter descriptions. The values of these parameters differ for each type of database node configured (server, client, or server with remote clients).

Not all parameters can be updated.

UPDATE DATABASE CONFIGURATION

Some 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. For more information on which parameters are configurable on-line and which ones are not, see summary list of configuration parameters.

For example, to change the *sortheap* database configuration parameter online for the SALES database, enter the following commands:

```
db2 connect to sales
db2 update db cfg using sortheap 1000
db2 connect reset
```

If an error occurs, the database configuration file does not change. The database configuration file cannot be updated if the checksum is invalid. This might 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.

Related concepts:

- “rah and db2_all commands overview” in *Administration Guide: Implementation*

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “db2CfgSet API - Set the database manager or database configuration parameters” in *Administrative API Reference*
- “Configuration parameters summary” in *Performance Guide*
- “GET DATABASE CONFIGURATION ” on page 457
- “RESET DATABASE CONFIGURATION ” on page 667
- “DBCFCG administrative view – Retrieve database configuration parameter information” in *Administrative SQL Routines and Views*
- “UPDATE DATABASE CONFIGURATION command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE DATABASE MANAGER CONFIGURATION

Modifies individual entries in the database manager configuration file.

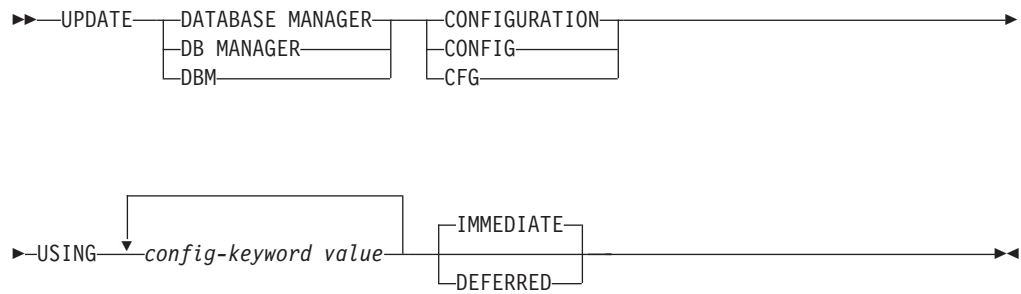
Authorization:

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. To update a configuration parameter on-line, it is also necessary to first attach to the instance.

Command syntax:



Command parameters:

DEFERRED

Make the changes only in the configuration file, so that the changes take effect when the instance is restarted.

IMMEDIATE

Make the changes right now, dynamically, while the instance is running. IMMEDIATE is the default, but it requires an instance attachment to be effective.

USING config-keyword value

Specifies the database manager configuration parameter to be updated. For a list of configuration parameters, refer to the configuration parameters summary.

Usage notes:

To view or print a list of the database manager configuration parameters, use the GET DATABASE MANAGER CONFIGURATION command. To reset the database manager configuration parameters to the recommended database manager defaults, use the RESET DATABASE MANAGER CONFIGURATION command. For more information about database manager configuration parameters and the values of these parameters appropriate for each type of database node configured (server, client, or server with remote clients), see individual configuration parameter descriptions.

Not all parameters can be updated.

UPDATE DATABASE MANAGER CONFIGURATION

Some changes to the database manager configuration file become effective only after they are loaded into memory. For more information on which parameters are configurable on-line and which ones are not, see the configuration parameter summary. Server configuration parameters that are not reset immediately are reset during execution of **db2start**. For a client configuration parameter, parameters are reset the next time you restart the application. If the client is the command line processor, it is necessary to invoke **TERMINATE**.

For example, to change the *DIAGLEVEL* database manager configuration parameter on-line for the **eastern** instance of the database manager, enter the following command:

```
db2 attach to eastern
db2 update dbm cfg using DIAGLEVEL 1
db2 detach
```

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 can occur if you edit database manager configuration file and do not use the appropriate command. If the checksum is invalid, you must reinstall the database manager to reset the database manager configuration file.

When you update the *SVCENAME*, or *TPNAME* database manager configuration parameters for the current instance, if LDAP support is enabled and there is an LDAP server registered for this instance, the LDAP server is updated with the new value or values.

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “db2CfgSet API - Set the database manager or database configuration parameters” in *Administrative API Reference*
- “Configuration parameters summary” in *Performance Guide*
- “GET DATABASE MANAGER CONFIGURATION ” on page 463
- “RESET DATABASE MANAGER CONFIGURATION ” on page 669
- “TERMINATE ” on page 744
- “DBMCFG administrative view – Retrieve database manager configuration parameter information” in *Administrative SQL Routines and Views*
- “UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE HEALTH NOTIFICATION CONTACT LIST

Updates the contact list for notification about health alerts issued by an instance.

Authorization:

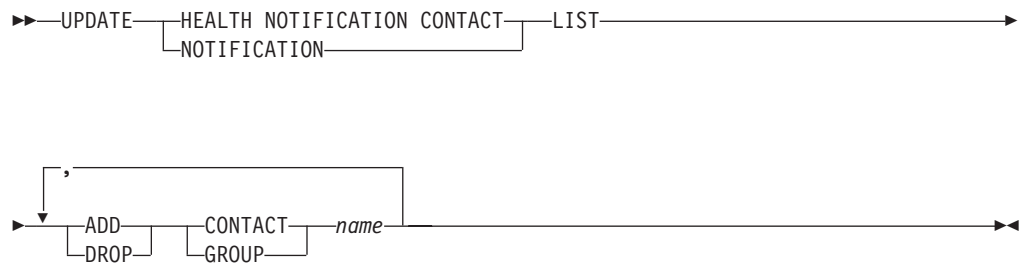
One of the following:

- sysadm
- sysctrl
- sysmaint

Required Connection:

Instance. An explicit attachment is not required.

Command Syntax:



Command Parameters:

ADD GROUP *name*

Add a new contact group that will notified of the health of the instance.

ADD CONTACT *name*

Add a new contact that will notified of the health of the instance.

DROP GROUP *name*

Removes the contact group from the list of contacts that will notified of the health of the instance.

DROP CONTACT *name*

Removes the contact from the list of contacts that will notified of the health of the instance.

Related tasks:

- “Enabling health alert notification” in *System Monitor Guide and Reference*

Related reference:

- “ADMIN_CMD procedure – Run administrative commands” in *Administrative SQL Routines and Views*
- “db2UpdateHealthNotificationList API - Update the list of contacts to whom health alert notifications can be sent” in *Administrative API Reference*
- “UPDATE HEALTH NOTIFICATION CONTACT LIST command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE HISTORY

Updates the location, device type, comment, or status in a history file entry.

Authorization:

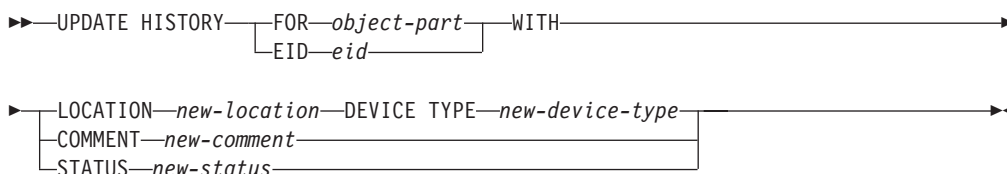
One of the following:

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

Required connection:

Database

Command syntax:



Command parameters:

FOR *object-part*

Specifies the identifier for the history entry to be updated. It is a time stamp with an optional sequence number from 001 to 999. This parameter cannot be used to update the entry status. To update the entry status, specify an EID instead.

EID *eid*

Specifies the history entry ID.

LOCATION *new-location*

Specifies the new physical location of a backup image. The interpretation of this parameter depends on the device type.

DEVICE TYPE *new-device-type*

Specifies a new device type for storing the backup image. Valid device types are:

| | |
|----------|---------------|
| D | Disk |
| K | Diskette |
| T | Tape |
| A | TSM |
| U | User exit |
| P | Pipe |
| N | Null device |
| X | XBSA |
| Q | SQL statement |

O Other**COMMENT** *new-comment*

Specifies a new comment to describe the entry.

STATUS *new-status*

Specifies a new status for an entry. Only backup entries can have their status updated. Valid values are:

- A** Active. Most entries are active.
- I** Inactive. Backup images that are no longer on the active log chain become inactive.
- E** Expired. Backup images that are no longer required because there are more than NUM_DB_BACKUPS active images are flagged as expired.
- D** Deleted. Backup images that are no longer available for recovery should be marked as having been deleted.

Example:

To update the history file entry for a 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 primary purpose of the database history file is to record information, but the data contained in the history is used directly by automatic restore operations. During any restore where the AUTOMATIC option is specified, the history of backup images and their locations will be referenced and used by the restore utility to fulfill the automatic restore request. If the automatic restore function is to be used and backup images have been relocated since they were created, it is recommended that the database history record for those images be updated to reflect the current location. If the backup image location in the database history is not updated, automatic restore will not be able to locate the backup images, but manual restore commands can still be used successfully.

Related concepts:

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “ADMIN_CMD procedure – Run administrative commands” in *Administrative SQL Routines and Views*
- “UPDATE HISTORY command using the ADMIN_CMD procedure” in *Administrative SQL Routines and Views*

UPDATE LDAP NODE

Updates the protocol information associated with a node entry that represents the DB2 server in Lightweight Directory Access Protocol (LDAP).

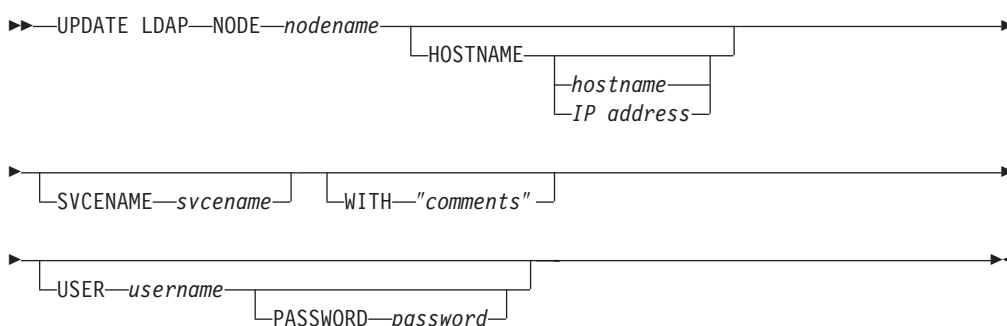
Authorization:

None

Required connection:

None

Command syntax:



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 or IP address

Specifies the TCP/IP host name or IP address.

- If it is a TCPIP node, the host name will be resolved to an IPv4 or IPv6 address.
- If it is a TCPIP4 node, the host name will be resolved to an IPv4 address only.
- If it is a TCPIP6 node, the host name will be resolved to an IPv6 address only.

SVCENAME svcename

Specifies the TCP/IP service name or port number.

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.

Related reference:

- “db2LdapUpdate API - Update the attributes of the DB2 server on the LDAP server” in *Administrative API Reference*
- “REGISTER ” on page 637
- “DEREGISTER ” on page 415

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 parameter.

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 the GET SNAPSHOT command reflects which, if any, switches are on.

Authorization:

One of the following:

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

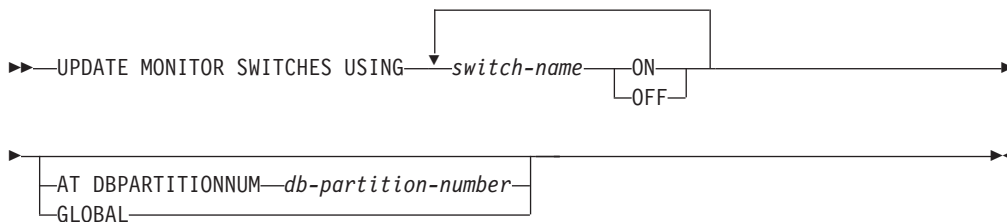
Required connection:

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:



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 |
| TIMESTAMP | Monitoring timestamp information |
| UOW | Unit of work information. |

AT DBPARTITIONNUM db-partition-number

Specifies the database partition for which the status of the monitor switches is to be displayed.

global Returns an aggregate result for all database partitions in a partitioned database system.

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 the GET MONITOR SWITCHES command.

Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODE can be substituted for DBPARTITIONNUM.

Related concepts:

- “System monitor switches” in *System Monitor Guide and Reference*

Related tasks:

- “Setting monitor switches from a client application” in *System Monitor Guide and Reference*

Related reference:

- “GET SNAPSHOT ” on page 487
- “GET MONITOR SWITCHES ” on page 478
- “db2MonitorSwitches API - Get or update the monitor switch settings” in *Administrative API Reference*

UPDATE MONITOR SWITCHES

Chapter 4. Using command line SQL statements and XQuery statements

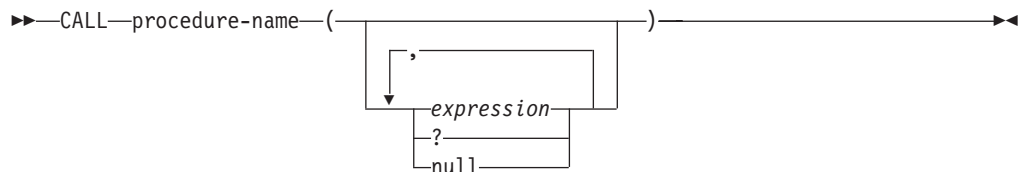
This section provides information about using Structured Query Language (SQL) statements from the command line. These statements can be executed directly from an operating system command prompt, and can be used to define and manipulate information stored in a database table, index, or view in much the 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.

You can use SQL statements from the command line, and you can use a stored procedure (SYSPROC.ADMIN_CMD()) to run some CLP commands through SQL. For more information on how to use this stored procedure, please refer to the SQL Administrative Routines.

To issue XQuery statements in CLP, prefix the statements with the XQUERY keyword.

All SQL statements that can be executed through the command line processor are listed in the CLP column of Table 15 on page 789. 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 CALL, CLOSE, CONNECT, DECLARE CURSOR, FETCH, and OPEN *does* depend on whether these statements are embedded or executed through the CLP. The CLP syntax of these statements is provided below:

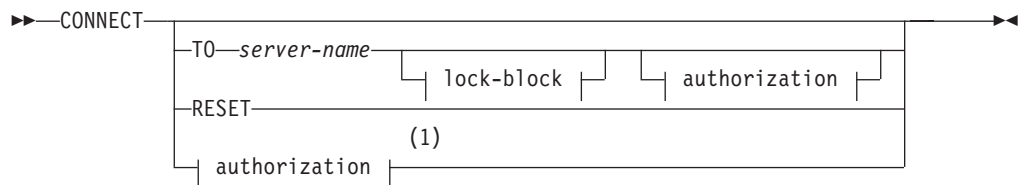
CALL



CLOSE

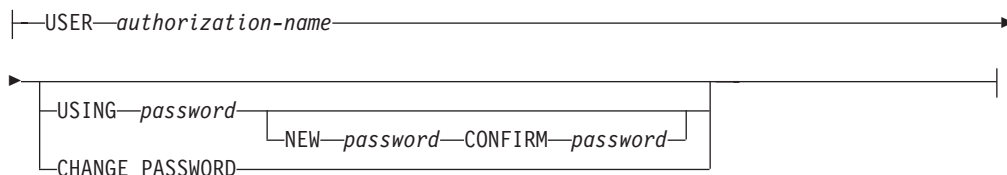


CONNECT

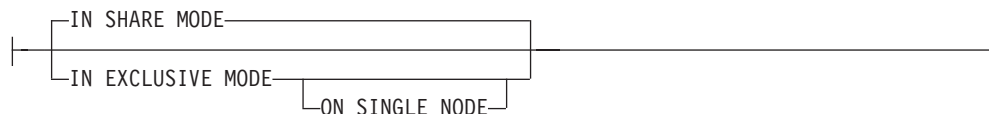


Using command line SQL statements and XQuery statements

authorization:



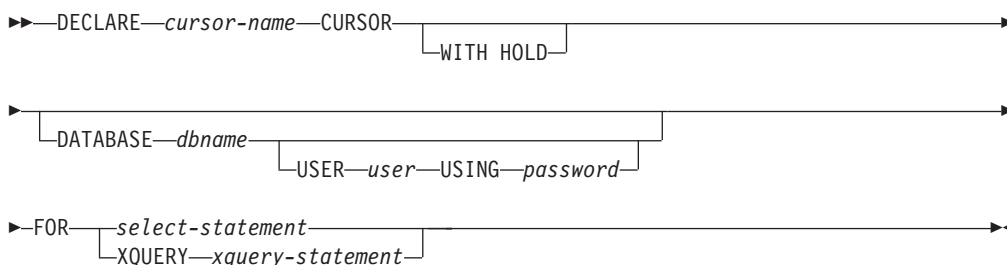
lock-block:



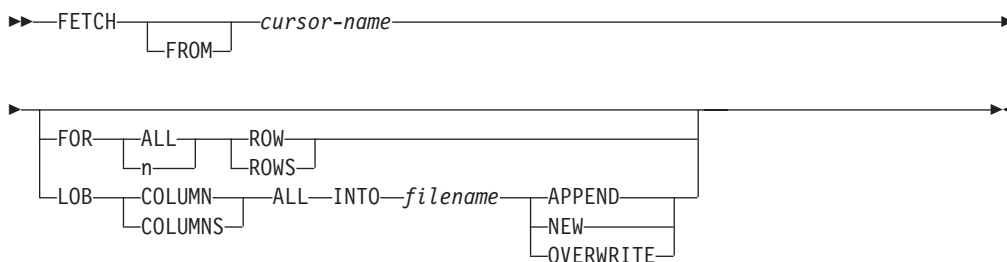
Notes:

- 1 This form is only valid if implicit connect is enabled.

DECLARE CURSOR



FETCH



OPEN



Notes:

1. When CALL is issued:
 - An expression must be used for each IN or INOUT parameter of the procedure. For an INOUT parameter, the expression must be a single literal value. The INOUT XML parameters must be either NULL (if nullable) or in the following format: XMLPARSE(DOCUMENT <*string*>). Note that the

Using command line SQL statements and XQuery statements

<string> in the argument for XMLPARSE must be a string literal and is subject to the CURRENT IMPLICIT XMLPARSE OPTION special register. It cannot be an expression.

- A question mark (?) must be used for each OUT parameter of the procedure.
- The stored procedure must be cataloged. If an uncataloged procedure is called, a SQL0440N error message is returned.

The following CLP script creates a procedure called PROC4 after it creates a table with an XML column C1. It uses three XML parameters: IN (PARM1), INOUT (PARM2) and OUT (PARM3), and returns a result set with XML data.

```
CREATE TABLE TAB4(C1 XML)
CREATE PROCEDURE PROC4(IN PARM1 XML, INOUT PARM2 XML, OUT PARM3 XML)
LANGUAGE SQL
BEGIN
    DECLARE STMT CLOB(1M) DEFAULT '';
    DECLARE C1 CURSOR WITH RETURN FOR S1;
    SET STMT = 'SELECT C1 FROM TAB4';

    /* INSERT PARM1 */
    INSERT INTO TAB4 VALUES(PARM1);

    /* MANIPULATE PARM2 */

    /* SET PARM3 AND INSERT */
    SET PARM3 = XMLPARSE(DOCUMENT '<a>333</a>');
    INSERT INTO TAB4 VALUES(PARM3);

    /* RETURN A RESULT SET WITH XML DATA */
    PREPARE S1 FROM STMT;
    OPEN C1;
END
```

To call the procedure PROC4 from the command line processor, issue a CALL statement:

```
CALL PROC4(XMLPARSE(DOCUMENT '<a>111</a>'), XMLPARSE(DOCUMENT '<a>222</a>'), ?)
```

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. The DATABASE clause in the DECLARE CURSOR statement is only applicable when the cursor is being used for a subsequent load from cursor operation.
4. To use the DECLARE CURSOR statement with an XQuery statement, users must prefix the XQuery statement with the keyword XQUERY explicitly.
5. When FETCH is issued through the command line processor, decimal and floating-point numbers are displayed with the territory's decimal delimiter, that is, a period (.) in the U.S., Canada, and the U.K.; a comma (,) in most

Using command line SQL statements and XQuery statements

other countries. However, when INSERT, UPDATE, CALL, 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.

6. When FETCH 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
```

7. A new LOB option has been added to FETCH. If the LOB clause is specified, only the next row is fetched:
 - When SELECT is issued through the command line processor to query tables containing LOB columns, all columns are truncated to 8KB in the output.
 - 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.
8. The command line processor displays BLOB columns in hexadecimal representation.
9. SQL statements that contain references to structured type columns cannot be issued if an appropriate transform function is not available.
10. A CLP imposed limit of 64K for SQL statements and for CLP commands that contain SQL statement components has now been removed.
11. XML data, retrieved via SELECT, CALL or XQuery, is truncated to 4000 bytes in the output.

To change the way that the CLP displays data (when querying databases using SQL statements through the CLP), rebind 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 +
db2clpr.r.bnd +
db2clpur.bnd +
db2clprs.bnd +
db2clpns.bnd
```

Using command line SQL statements and XQuery statements

2. Connect to the database.
3. Issue the following command:

```
db2 bind @c1p.lst collection nullid datetime iso
```

Table 15. SQL Statements (DB2)

| SQL Statement | Dynamic ¹ | Command Line Processor (CLP) | Call Level Interface ³ (CLI) | SQL Procedure |
|---|----------------------|------------------------------|---|-----------------|
| ALLOCATE CURSOR | | | | X |
| assignment statement | | | | X |
| ASSOCIATE LOCATORS | | | | X |
| ALTER { BUFFERPOOL, NICKNAME, ⁹ NODEGROUP, SERVER, ⁹ TABLE, TABLESPACE, USER MAPPING, ⁹ TYPE, VIEW } | X | X | X | |
| BEGIN DECLARE SECTION ² | | | | |
| CALL | X | X | X | X |
| CASE statement | | | | X |
| CLOSE | | X | SQLCloseCursor(), SQLFreeStmt() | X |
| COMMENT ON | X | X | X | X |
| COMMIT | X | X | SQLEndTran(), SQLTransact() | X |
| Compound SQL (Embedded) | | | X ⁴ | |
| compound statement | | | | X |
| CONNECT (Type 1) | | X | SQLBrowseConnect(), SQLConnect(), SQLDriverConnect() | |
| CONNECT (Type 2) | | X | SQLBrowseConnect(), SQLConnect(), SQLDriverConnect() | |
| CREATE { ALIAS, BUFFERPOOL, DISTINCT TYPE, EVENT MONITOR, FUNCTION, FUNCTION MAPPING, ⁹ INDEX, INDEX EXTENSION, METHOD, NICKNAME, ⁹ NODEGROUP, PROCEDURE, SCHEMA, SERVER, TABLE, TABLESPACE, TRANSFORM, TYPE MAPPING, ⁹ TRIGGER, USER MAPPING, ⁹ TYPE, VIEW, WRAPPER ⁹ } | X | X | X | X ¹⁰ |
| DECLARE CURSOR ² | | X | SQLAllocStmt() | X |
| DECLARE GLOBAL TEMPORARY TABLE | X | X | X | X |
| DELETE | X | X | X | X |
| DESCRIBE ⁸ | | X | SQLColAttributes(), SQLDescribeCol(), SQLDescribeParam() ⁶ | |
| DISCONNECT | | X | SQLDisconnect() | |
| DROP | X | X | X | X ¹⁰ |
| END DECLARE SECTION ² | | | | |

Using command line SQL statements and XQuery statements

Table 15. SQL Statements (DB2) (continued)

| SQL Statement | Dynamic ¹ | Command Line Processor (CLP) | Call Level Interface ³ (CLI) | SQL Procedure |
|-------------------------------------|----------------------|------------------------------|--|---------------|
| EXECUTE | | | SQLExecute() | X |
| EXECUTE IMMEDIATE | | | SQLExecDirect() | X |
| EXPLAIN | X | X | X | X |
| FETCH | | X | SQLExtendedFetch(), SQLFetch(), SQLFetchScroll() | X |
| FLUSH EVENT MONITOR | X | X | X | |
| FOR statement | | | | X |
| FREE LOCATOR | | | X ⁴ | X |
| GET DIAGNOSTICS | | | | X |
| GOTO statement | | | | X |
| GRANT | X | X | X | X |
| IF statement | | | | X |
| INCLUDE ² | | | | |
| INSERT | X | X | X | X |
| ITERATE | | | | X |
| LEAVE statement | | | | X |
| LOCK TABLE | X | X | X | X |
| LOOP statement | | | | X |
| OPEN | | X | SQLExecute(), SQLExecDirect() | X |
| PREPARE | | | SQLPrepare() | X |
| REFRESH TABLE | X | X | X | |
| RELEASE | | X | | X |
| RELEASE SAVEPOINT | X | X | X | X |
| RENAME TABLE | X | X | X | |
| RENAME TABLESPACE | X | X | X | |
| REPEAT statement | | | | X |
| RESIGNAL statement | | | | X |
| RETURN statement | | | | X |
| REVOKE | X | X | X | |
| ROLLBACK | X | X | SQLEndTran(), SQLTransact() | X |
| SAVEPOINT | X | X | X | X |
| select-statement | X | X | X | X |
| SELECT INTO | | | | X |
| SET CONNECTION | | X | SQLSetConnection() | |
| SET CURRENT DEFAULT TRANSFORM GROUP | X | X | X | X |
| SET CURRENT DEGREE | X | X | X | X |
| SET CURRENT EXPLAIN MODE | X | X | X, SQLSetConnectAttr() | X |

Using command line SQL statements and XQuery statements

Table 15. SQL Statements (DB2) (continued)

| SQL Statement | Dynamic ¹ | Command Line Processor (CLP) | Call Level Interface ³ (CLI) | SQL Procedure |
|--------------------------------------|----------------------|------------------------------|---|---------------|
| SET CURRENT EXPLAIN SNAPSHOT | X | X | X, SQLSetConnectAttr() | X |
| SET CURRENT PACKAGESET | | | | |
| SET CURRENT QUERY OPTIMIZATION | X | X | X | X |
| SET CURRENT REFRESH AGE | X | X | X | X |
| SET EVENT MONITOR STATE | X | X | X | X |
| SET INTEGRITY | X | X | X | |
| SET PASSTHRU ⁹ | X | X | X | X |
| SET PATH | X | X | X | X |
| SET SCHEMA | X | X | X | X |
| SET SERVER OPTION ⁹ | X | X | X | X |
| SET transition-variable ⁵ | X | X | X | X |
| SIGNAL statement | | | | X |
| SIGNAL SQLSTATE ⁵ | X | X | X | |
| UPDATE | X | X | X | X |
| VALUES INTO | | | | X |
| WHENEVER ² | | | | |
| WHILE statement | | | | X |

Notes:

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.
9. Statement is supported only for federated database servers.
10. SQL procedures can only issue CREATE and DROP statements for indexes, tables, and views.

Related reference:

- “CONNECT (Type 2) statement” in *SQL Reference, Volume 2*
- “CLOSE statement” in *SQL Reference, Volume 2*
- “CONNECT (Type 1) statement” in *SQL Reference, Volume 2*
- “DECLARE CURSOR statement” in *SQL Reference, Volume 2*
- “FETCH statement” in *SQL Reference, Volume 2*
- “OPEN statement” in *SQL Reference, Volume 2*
- “SELECT statement” in *SQL Reference, Volume 2*

Using command line SQL statements and XQuery statements

Appendix A. How to read the syntax diagrams

Throughout this book, syntax is described using the structure defined as follows:

Read the syntax diagrams from left to right and top to bottom, following the path of the line.

The \blacktriangleright — symbol indicates the beginning of a syntax diagram.

The — \blacktriangleright symbol indicates that the syntax is continued on the next line.

The \blacktriangleright — symbol indicates that the syntax is continued from the previous line.

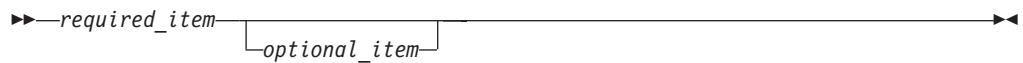
The — \blacktriangleleft symbol indicates the end of a syntax diagram.

Syntax fragments start with the |— symbol and end with the —| symbol.

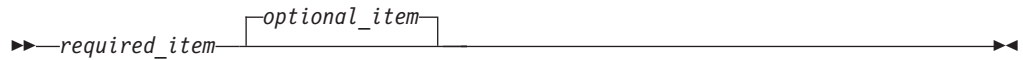
Required items appear on the horizontal line (the main path).



Optional items appear below the main path.



If an optional item appears above the main path, that item has no effect on execution, and is used only for readability.

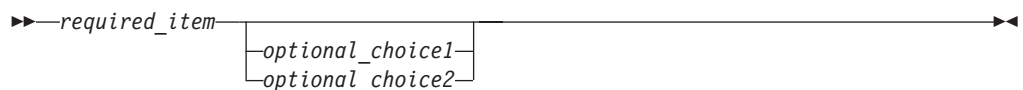


If you can choose from two or more items, they appear in a stack.

If you *must* choose one of the items, one item of the stack appears on the main path.

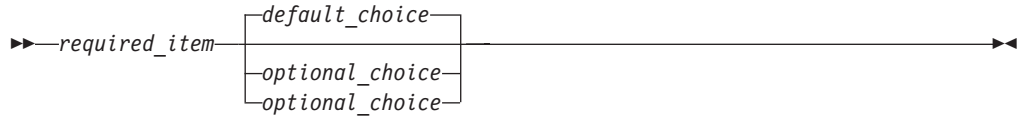


If choosing one of the items is optional, the entire stack appears below the main path.



How to read the syntax diagrams

If one of the items is the default, it will appear above the main path, and the remaining choices will be shown below.



An arrow returning to the left, above the main line, indicates an item that can be repeated. In this case, repeated items must be separated by one or more blanks.



If the repeat arrow contains a comma, you must separate repeated items with a comma.



A repeat arrow above a stack indicates that you can make more than one choice from the stacked items or repeat a single choice.

Keywords appear in uppercase (for example, FROM). They must be spelled exactly as shown. Variables appear in lowercase (for example, column-name). They represent user-supplied names or values in the syntax.

If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Sometimes a single variable represents a larger fragment of the syntax. For example, in the following diagram, the variable `parameter-block` represents the whole syntax fragment that is labeled **parameter-block**:



parameter-block:



Adjacent segments occurring between “large bullets” (●) may be specified in any sequence.



How to read the syntax diagrams

The above diagram shows that item2 and item3 may be specified in either order. Both of the following are valid:

```
required_item item1 item2 item3 item4  
required_item item1 item3 item2 item4
```

How to read the syntax diagrams

Appendix B. Naming conventions

The following conventions apply when naming database manager objects, such as databases and tables:

- Character strings that represent names of database manager objects can contain any of the following: a-z, A-Z, 0-9, @, #, and \$.
- 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 convention is character strings that represent names under the systems network architecture (SNA). Many values are case sensitive, such as logical unit names (partner_lu and local_lu). 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. The schema name for a declared temporary table must be SESSION.

- Local aliases for remote nodes that are to be cataloged in the node directory cannot exceed eight characters in length.
- 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.

The following conventions apply when naming user IDs 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 \$.
- User IDs and groups may also contain any of the following additional characters when supported by the security plug-in: `_ !, %, (,), {, }, -, ., ^`.
- User IDs and groups containing any of the following characters must be delimited with quotations when entered through the command line processor: `!, %, (,), {, }, -, ., ^`.
- 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.
- Authentication IDs cannot exceed 30 characters on Windows 32-bit operating systems and 8 characters on all other operating systems.
- Group IDs cannot exceed 30 characters in length.

Related reference:

- "CREATE DATABASE " on page 395
- "CREATE TOOLS CATALOG " on page 408

Appendix C. File type modifiers

The following topics describe the file type modifiers for the load, import and export utilities.

File type modifiers for the load utility

Table 16. Valid file type modifiers for the load utility: All file formats

| Modifier | Description |
|-------------------|---|
| anyorder | This modifier is used in conjunction with the <i>cpu_parallelism</i> parameter. Specifies that the preservation of source data order is not required, yielding significant additional performance benefit on SMP systems. If the value of <i>cpu_parallelism</i> is 1, this option is ignored. This option is not supported if <i>SAVECOUNT</i> > 0, since crash recovery after a consistency point requires that data be loaded in sequence. |
| generatedignore | This modifier informs the load utility that data for all generated columns is present in the data file but should be ignored. This results in all generated column values being generated by the utility. This modifier cannot be used with either the <i>generatedmissing</i> or the <i>generatedoverride</i> modifier. |
| generatedmissing | If this modifier is specified, the utility assumes that the input data file contains no data for the generated column (not even NULLs). This results in all generated column values being generated by the utility. This modifier cannot be used with either the <i>generatedignore</i> or the <i>generatedoverride</i> modifier. |
| generatedoverride | <p>This modifier instructs the load utility to accept user-supplied data for all generated columns in the table (contrary to the normal rules for these types of columns). This is useful when migrating data from another database system, or when loading a table from data that was recovered using the RECOVER DROPPED TABLE option on the ROLLFORWARD DATABASE command. When this modifier is used, any rows with no data or NULL data for a non-nullable generated column will be rejected (SQL3116W). When this modifier is used, the table will be placed in Set Integrity Pending state. To take the table out of Set Integrity Pending state without verifying the user-supplied values, issue the following command after the load operation:</p> <pre>SET INTEGRITY FOR < table-name > GENERATED COLUMN IMMEDIATE UNCHECKED</pre> <p>To take the table out of Set Integrity Pending state and force verification of the user-supplied values, issue the following command after the load operation:</p> <pre>SET INTEGRITY FOR < table-name > IMMEDIATE CHECKED.</pre> <p>When this modifier is specified and there is a generated column in any of the partitioning keys, dimension keys or distribution keys, then the LOAD command will automatically convert the modifier to <i>generatedignore</i> and proceed with the load. This will have the effect of regenerating all of the generated column values.</p> <p>This modifier cannot be used with either the <i>generatedmissing</i> or the <i>generatedignore</i> modifier.</p> |
| identityignore | This modifier informs the load utility that data for the identity column is present in the data file but should be ignored. This results in all identity values being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with either the <i>identitymissing</i> or the <i>identityoverride</i> modifier. |

File type modifiers for the load utility

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

| Modifier | Description |
|--------------------------|--|
| identitymissing | If this modifier is specified, the utility assumes that the input data file contains no data for the identity column (not even NULLs), and will therefore generate a value for each row. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This modifier cannot be used with either the identityignore or the identityoverride modifier. |
| identityoverride | This modifier should be used only when an identity column defined as GENERATED ALWAYS is present in the table to be loaded. It instructs the utility to accept explicit, non-NULL data for such a column (contrary to the normal rules for these types of identity columns). This is useful when migrating data from another database system when the table must be defined as GENERATED ALWAYS, or when loading a table from data that was recovered using the DROPPED TABLE RECOVERY option on the ROLLFORWARD DATABASE command. When this modifier is used, any rows with no data or NULL data for the identity column will be rejected (SQL3116W). This modifier cannot be used with either the identitymissing or the identityignore modifier. The load utility will not attempt to maintain or verify the uniqueness of values in the table's identity column when this option is used. |
| indexfreespace= <i>x</i> | <p><i>x</i> 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 load rebuilds the index. Load with INDEXING MODE INCREMENTAL ignores this option. The first entry in a page is added without restriction; subsequent entries are added the percent free space threshold can be maintained. The default value is the one used at CREATE INDEX time.</p> <p>This value takes precedence over the PCTFREE value specified in the CREATE INDEX statement; the registry variable DB2 INDEX FREE takes precedence over indexfreespace. The indexfreespace option affects index leaf pages only.</p> |
| lobsinfile | <p><i>lob-path</i> specifies the path to the files containing LOB data. The ASC, DEL, or IXF load input files contain the names of the files having LOB data in the LOB column.</p> <p>This option is not supported in conjunction with the CURSOR filetype.</p> <p>The LOBS FROM clause specifies where the LOB files are located when the "lobsinfile" modifier is used. The LOBS FROM clause will implicitly activate the LOBSINFILE behavior. The LOBS FROM clause conveys to the LOAD utility the list of paths to search for the LOB files while loading the data.</p> <p>Each path contains at least one file that contains at least one LOB pointed to by a Lob Location Specifier (LLS) in the data file. The LLS is a string representation of the location of a LOB in a file stored in the LOB file path. The format of an LLS is <i>filename.ext.nnn.mmm/</i>, where <i>filename.ext</i> is the name of the file that contains the LOB, <i>nnn</i> is the offset in bytes of the LOB within the file, and <i>mmm</i> is the length of the LOB in bytes. For example, if the string <i>db2exp.001.123.456/</i> is stored in the data file, the LOB is located at offset 123 in the file <i>db2exp.001</i>, and is 456 bytes long.</p> <p>To indicate a null LOB, enter the size as -1. If the size is specified as 0, it is treated as a 0 length LOB. For null LOBS with length of -1, the offset and the file name are ignored. For example, the LLS of a null LOB might be <i>db2exp.001.7.-1/</i>.</p> |

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

| Modifier | Description |
|-------------------------|--|
| noheader | <p>Skips the header verification code (applicable only to load operations into tables that reside in a single-partition database partition group).</p> <p>If the default MPP load (mode PARTITION_AND_LOAD) is used against a table residing in a single-partition database partition group, the file is not expected to have a header. Thus the noheader modifier is not needed. If the LOAD_ONLY mode is used, the file is expected to have a header. The only circumstance in which you should need to use the noheader modifier is if you wanted to perform LOAD_ONLY operation using a file that does not have a header.</p> |
| norowwarnings | <p>Suppresses all warnings about rejected rows.</p> |
| pagefreespace= <i>x</i> | <p><i>x</i> 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. 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 <i>x</i> 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. 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.</p> |
| seclabelchar | <p>Indicates that security labels in the input source file are in the string format for security label values rather than in the default encoded numeric format. LOAD converts each security label into the internal format as it is loaded. If a string is not in the proper format the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3242W) is returned. If the string does not represent a valid security label that is part of the security policy protecting the table then the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3243W) is returned.</p> <p>This modifier cannot be specified if the seclabelname modifier is specified, otherwise the load fails and an error (SQLCODE SQL3525N) is returned.</p> <p>If you have a table consisting of a single DB2SECURITYLABEL column, the data file might look like this:</p> <pre> "CONFIDENTIAL:ALPHA:G2" "CONFIDENTIAL;SIGMA:G2" "TOP SECRET:ALPHA:G2" </pre> <p>To load or import this data, the SECLABELCHAR file type modifier must be used:</p> <pre> LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1 </pre> |

File type modifiers for the load utility

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

| Modifier | Description |
|-------------------------|--|
| seclabelname | <p>Indicates that security labels in the input source file are indicated by their name rather than the default encoded numeric format. LOAD will convert the name to the appropriate security label if it exists. If no security label exists with the indicated name for the security policy protecting the table the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3244W) is returned.</p> <p>This modifier cannot be specified if the seclabelchar modifier is specified, otherwise the load fails and an error (SQLCODE SQL3525N) is returned.</p> <p>If you have a table consisting of a single DB2SECURITYLABEL column, the data file might consist of security label names similar to:</p> <pre>"LABEL1" "LABEL1" "LABEL2"</pre> <p>To load or import this data, the SECLABELNAME file type modifier must be used:</p> <pre>LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1</pre> <p>Note: If the file type is ASC, any spaces following the name of the security label will be interpreted as being part of the name. To avoid this use the striptblanks file type modifier to make sure the spaces are removed.</p> |
| totalreespace= <i>x</i> | <p><i>x</i> is an integer greater than or equal to 0 . 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 <i>x</i> is 20, and the table has 100 data pages after the data has been loaded, 20 additional empty pages will be appended. The total number of data pages for the table will be 120. The data pages total does not factor in the number of index pages in the table. This option does not affect the index object. If two loads are done with this option specified, the second load will not reuse the extra space appended to the end by the first load.</p> |
| usedefaults | <p>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:</p> <ul style="list-style-type: none"> • For DEL files: two adjacent column delimiters (",,") or two adjacent column delimiters separated by an arbitrary number of spaces (" , ") are specified for a column value. • For DEL/ASC/WSF files: A row that does not have enough columns, or is not long enough for the original specification. For ASC files, NULL column values are not considered explicitly missing, and a default will not be substituted for NULL column values. NULL column values are represented by all space characters for numeric, date, time, and /timestamp columns, or by using the NULL INDICATOR for a column of any type to indicate the column is NULL. <p>Without this option, if a source column contains no data for a row instance, one of the following occurs:</p> <ul style="list-style-type: none"> • For DEL/ASC/WSF files: If the column is nullable, a NULL is loaded. If the column is not nullable, the utility rejects the row. |

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL)

| Modifier | Description |
|-------------------------|---|
| codepage= <i>x</i> | <p><i>x</i> 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.</p> <p>The following rules apply:</p> <ul style="list-style-type: none"> • For pure DBCS (graphic), mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive. • For DEL data specified in an EBCDIC code page, the delimiters might not coincide with the shift-in and shift-out DBCS characters. • 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. <p>This option is not supported in conjunction with the CURSOR filetype.</p> |
| dateformat=" <i>x</i> " | <p><i>x</i> is the format of the date in the source file.¹ Valid date elements are:</p> <p>YYYY - Year (four digits ranging from 0000 - 9999)
 M - Month (one or two digits ranging from 1 - 12)
 MM - Month (two digits ranging from 1 - 12;
 mutually exclusive with M)
 D - Day (one or two digits ranging from 1 - 31)
 DD - Day (two digits ranging from 1 - 31;
 mutually exclusive with D)
 DDD - Day of the year (three digits ranging
 from 001 - 366; mutually exclusive
 with other day or month elements)</p> <p>A default value of 1 is assigned for each element that is not specified. Some examples of date formats are:</p> <p>"D-M-YYYY"
 "MM.DD.YYYY"
 "YYYYDDD"</p> |
| dumpfile = <i>x</i> | <p><i>x</i> is the fully qualified (according to the server database partition) name of an exception file to which rejected rows are written. A maximum of 32 KB of data is written per record. Following is an example that shows how to specify a dump file:</p> <pre>db2 load from data of del modified by dumpfile = /u/user/filename insert into table_name</pre> <p>The file will be created and owned by the instance owner. To override the default file permissions, use the dumpfileaccessall file type modifier.</p> <p>Notes:</p> <ol style="list-style-type: none"> 1. In a partitioned database environment, the path should be local to the loading database partition, so that concurrently running load operations do not attempt to write to the same file. 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. |

File type modifiers for the load utility

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

| Modifier | Description |
|-------------------|--|
| dumpfileaccessall | <p>Grants read access to 'OTHERS' when a dump file is created.</p> <p>This file type modifier is only valid when:</p> <ol style="list-style-type: none"> 1. it is used in conjunction with dumpfile file type modifier 2. the user has SELECT privilege on the load target table 3. it is issued on a DB2 server database partition that resides on a UNIX operating system |
| fastparse | <p>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.</p> |
| implieddecimal | <p>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, <i>not</i> 12345.00.</p> <p>This modifier cannot be used with the packeddecimal modifier.</p> |
| timeformat="x" | <p>x is the format of the time in the source file.¹ Valid time elements are:</p> <ul style="list-style-type: none"> H - Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system) HH - Hour (two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system; mutually exclusive with H) M - Minute (one or two digits ranging from 0 - 59) MM - Minute (two digits ranging from 0 - 59; mutually exclusive with M) S - Second (one or two digits ranging from 0 - 59) SS - Second (two digits ranging from 0 - 59; mutually exclusive with S) SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86399; mutually exclusive with other time elements) TT - Meridian indicator (AM or PM) <p>A default value of 0 is assigned for each element that is not specified. Some examples of time formats are:</p> <pre>"HH:MM:SS" "HH.MM TT" "SSSSS"</pre> |

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

| Modifier | Description |
|---------------------|--|
| timestampformat="x" | <p>x is the format of the time stamp in the source file.¹ Valid time stamp elements are:</p> <p>YYYY - Year (four digits ranging from 0000 - 9999)</p> <p>M - Month (one or two digits ranging from 1 - 12)</p> <p>MM - Month (two digits ranging from 01 - 12; mutually exclusive with M and MMM)</p> <p>MMM - Month (three-letter case-insensitive abbreviation for the month name; mutually exclusive with M and MM)</p> <p>D - Day (one or two digits ranging from 1 - 31)</p> <p>DD - Day (two digits ranging from 1 - 31; mutually exclusive with D)</p> <p>DDD - Day of the year (three digits ranging from 001 - 366; mutually exclusive with other day or month elements)</p> <p>H - Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system)</p> <p>HH - Hour (two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system; mutually exclusive with H)</p> <p>M - Minute (one or two digits ranging from 0 - 59)</p> <p>MM - Minute (two digits ranging from 0 - 59; mutually exclusive with M, minute)</p> <p>S - Second (one or two digits ranging from 0 - 59)</p> <p>SS - Second (two digits ranging from 0 - 59; mutually exclusive with S)</p> <p>SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86399; mutually exclusive with other time elements)</p> <p>UUUUUU - Microsecond (6 digits ranging from 000000 - 999999; mutually exclusive with all other microsecond elements)</p> <p>UUUUU - Microsecond (5 digits ranging from 00000 - 99999, maps to range from 000000 - 999990; mutually exclusive with all other microsecond elements)</p> <p>UUUU - Microsecond (4 digits ranging from 0000 - 9999, maps to range from 000000 - 999900; mutually exclusive with all other microsecond elements)</p> <p>UUU - Microsecond (3 digits ranging from 000 - 999, maps to range from 000000 - 999000; mutually exclusive with all other microsecond elements)</p> <p>UU - Microsecond (2 digits ranging from 00 - 99, maps to range from 000000 - 990000; mutually exclusive with all other microsecond elements)</p> <p>U - Microsecond (1 digit ranging from 0 - 9, maps to range from 000000 - 900000; mutually exclusive with all other microsecond elements)</p> <p>TT - Meridian indicator (AM or PM)</p> <p>A default value of 1 is assigned for unspecified YYYY, M, MM, D, DD, or DDD elements. A default value of 'Jan' is assigned to an unspecified MMM element. A default value of 0 is assigned for all other unspecified elements. Following is an example of a time stamp format:</p> <pre>"YYYY/MM/DD HH:MM:SS.UUUUUU"</pre> <p>The valid values for the MMM element include: 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' and 'dec'. These values are case insensitive.</p> <p>The following example illustrates how to import data containing user defined date and time formats into a table called schedule:</p> <pre>db2 import from delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule</pre> |

File type modifiers for the load utility

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

| Modifier | Description |
|--------------------|---|
| usegraphiccodepage | <p>If usegraphiccodepage is given, the assumption is made that data being loaded into graphic or double-byte character large object (DBCLOB) data field(s) is in the graphic code page. The rest of the data is assumed to be in the character code page. The graphic codepage is associated with the character code page. LOAD determines the character code page through either the codepage modifier, if it is specified, or through the code page of the database if the codepage modifier is not specified.</p> <p>This modifier should be used in conjunction with the delimited data file generated by drop table recovery only if the table being recovered has graphic data.</p> <p>Restrictions</p> <p>The usegraphi ccodepage modifier MUST NOT be specified with DEL files created by the EXPORT utility, as these files contain data encoded in only one code page. The usegraphi ccodepage modifier is also ignored by the double-byte character large objects (DBCLOBs) in files.</p> |

Table 18. Valid file type modifiers for the load utility: ASC file formats (Non-delimited ASCII)

| Modifier | Description |
|----------------|--|
| binarynumerics | <p>Numeric (but not DECIMAL) data must be in binary form, not the character representation. This avoids costly conversions.</p> <p>This option is supported only with positional ASC, using fixed length records specified by the reclen option.</p> <p>The following rules apply:</p> <ul style="list-style-type: none"> • No conversion between data types is performed, with the exception of BIGINT, INTEGER, and SMALLINT. • Data lengths must match their target column definitions. • FLOATs must be in IEEE Floating Point format. • Binary data in the load source file is assumed to be big-endian, regardless of the platform on which the load operation is running. <p>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.</p> |
| nochecklengths | <p>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.</p> |
| nullindchar=x | <p>x is a single character. Changes the character denoting a NULL value to x. The default value of x is Y.²</p> <p>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.</p> |

Table 18. Valid file type modifiers for the load utility: ASC file formats (Non-delimited ASCII) (continued)

| Modifier | Description |
|------------------|--|
| packeddecimal | <p>Loads packed-decimal data directly, since the <code>binarynumerics</code> modifier does not include the <code>DECIMAL</code> field type.</p> <p>This option is supported only with positional ASC, using fixed length records specified by the <code>reclen</code> option.</p> <p>Supported values for the sign nibble are:</p> <ul style="list-style-type: none"> + = 0xC 0xA 0xE 0xF - = 0xD 0xB <p>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.</p> <p>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 Windows operating systems, the byte order must not be reversed.</p> <p>This modifier cannot be used with the <code>implieddecimal</code> modifier.</p> |
| reclen= <i>x</i> | <p><i>x</i> is an integer with a maximum value of 32 767. <i>x</i> characters are read for each row, and a new-line character is not used to indicate the end of the row.</p> |
| striptblanks | <p>Truncates any trailing blank spaces when loading data into a variable-length field. If this option is not specified, blank spaces are kept.</p> <p>This option cannot be specified together with <code>striptnulls</code>. These are mutually exclusive options. This option replaces the obsolete <code>t</code> option, which is supported for back-level compatibility only.</p> |
| striptnulls | <p>Truncates any trailing NULLs (0x00 characters) when loading data into a variable-length field. If this option is not specified, NULLs are kept.</p> <p>This option cannot be specified together with <code>striptblanks</code>. These are mutually exclusive options. This option replaces the obsolete <code>padwithzero</code> option, which is supported for back-level compatibility only.</p> |
| zoneddecimal | <p>Loads zoned decimal data, since the <code>BINARYNUMERICS</code> modifier does not include the <code>DECIMAL</code> field type. This option is supported only with positional ASC, using fixed length records specified by the <code>RECLLEN</code> option.</p> <p>Half-byte sign values can be one of the following:</p> <ul style="list-style-type: none"> + = 0xC 0xA 0xE 0xF - = 0xD 0xB <p>Supported values for digits are 0x0 to 0x9.</p> <p>Supported values for zones are 0x3 and 0xF.</p> |

File type modifiers for the load utility

Table 19. Valid file type modifiers for the load utility: DEL file formats (Delimited ASCII)

| Modifier | Description |
|-----------------|--|
| chardelx | <p>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.²³ If you wish to explicitly specify the double quotation mark(") as the character string delimiter, you should specify it as follows:</p> <pre>modified by chardel""</pre> <p>The single quotation mark (') can also be specified as a character string delimiter as follows:</p> <pre>modified by chardel''</pre> |
| coldelx | <p>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.²³</p> |
| decplusblank | <p>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.</p> |
| decptx | <p>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.²³</p> |
| delprioritychar | <p>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:</p> <pre>db2 load ... modified by delprioritychar ...</pre> <p>For example, given the following DEL data file:</p> <pre>"Smith, Joshua",4000,34.98<row delimiter> "Vincent,<row delimiter>, is a manager", 4005,44.37<row delimiter></pre> <p>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 <i>not</i> specified, there will be three rows in this data file, each delimited by a <row delimiter>.</p> |
| keepblanks | <p>Preserves the leading and trailing blanks in each field of type CHAR, VARCHAR, LONG VARCHAR, or CLOB. Without this option, all leading and trailing blanks that are not inside character delimiters are removed, and a NULL is inserted into the table for all blank fields.</p> <p>The following example illustrates how to load data into a table called TABLE1, while preserving all leading and trailing spaces in the data file:</p> <pre>db2 load from delfile3 of del modified by keepblanks insert into table1</pre> |
| nochardel | <p>The load utility will assume all bytes found between the column delimiters to be part of the column's data. Character delimiters will be parsed as part of column data. This option should not be specified if the data was exported using DB2 (unless nochardel was specified at export time). It is provided to support vendor data files that do not have character delimiters. Improper usage might result in data loss or corruption.</p> <p>This option cannot be specified with chardelx, delprioritychar or nodoubledel. These are mutually exclusive options.</p> |
| nodoubledel | <p>Suppresses recognition of double character delimiters.</p> |

Table 20. Valid file type modifiers for the load utility: IXF file format

| Modifier | Description |
|----------------|--|
| forcein | Directs the utility to accept data despite code page mismatches, and to suppress translation between code pages.

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. Double quotation marks around the date format string are mandatory. Field separators cannot contain any of the following: a-z, A-Z, and 0-9. The field separator should not be the same as the character delimiter or field delimiter in the DEL file format. A field separator is optional if the start and end positions of an element are unambiguous. Ambiguity can exist if (depending on the modifier) elements such as D, H, M, or S are used, because of the variable length of the entries.

For time stamp formats, care must be taken to avoid ambiguity between the month and the minute descriptors, since they both use the letter M. A month field must be adjacent to other date fields. A minute field must be adjacent to other time fields. Following are some ambiguous time stamp formats:

```
"M" (could be a month, or a minute)
"M:M" (Which is which?)
"M:YYYY:M" (Both are interpreted as month.)
"S:M:YYYY" (adjacent to both a time value and a date value)
```

In ambiguous cases, the utility will report an error message, and the operation will fail.

Following are some unambiguous time stamp formats:

```
"M:YYYY" (Month)
"S:M" (Minute)
"M:YYYY:S:M" (Month...Minute)
"M:H:YYYY:M:D" (Minute...Month)
```

Some characters, such as double quotation marks and back slashes, must be preceded by an escape character (for example, \).

2. 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 ...
```
3. Delimiter restrictions for moving data lists restrictions that apply to the characters that can be used as delimiter overrides.

File type modifiers for the load utility

- 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.

Table 21. LOAD behavior when using codepage and usegraphiccodepage

| codepage=N | usegraphiccodepage | LOAD behavior |
|------------|--------------------|---|
| Absent | Absent | All data in the file is assumed to be in the database code page, not the application code page, even if the CLIENT option is specified. |
| Present | Absent | All data in the file is assumed to be in code page N.

Warning: Graphic data will be corrupted when loaded into the database if N is a single-byte code page. |
| Absent | Present | Character data in the file is assumed to be in the database code page, even if the CLIENT option is specified. Graphic data is assumed to be in the code page of the database graphic data, even if the CLIENT option is specified.

If the database code page is single-byte, then all data is assumed to be in the database code page.

Warning: Graphic data will be corrupted when loaded into a single-byte database. |
| Present | Present | Character data is assumed to be in code page N. Graphic data is assumed to be in the graphic code page of N.

If N is a single-byte or double-byte code page, then all data is assumed to be in code page N.

Warning: Graphic data will be corrupted when loaded into the database if N is a single-byte code page. |

Related reference:

- “db2Load API - Load data into a table” in *Administrative API Reference*
- “Delimiter restrictions for moving data” on page 826
- “LOAD ” on page 557

File type modifiers for the import utility

Table 22. Valid file type modifiers for the import utility: All file formats

| Modifier | Description |
|------------|---|
| compound=x | <p>x is a number between 1 and 100 inclusive. Uses nonatomic compound SQL to insert the data, and x statements will be attempted each time.</p> <p>If this modifier is specified, and the transaction log is not sufficiently large, the import operation will fail. The transaction log must be large enough to accommodate either the number of rows specified by COMMITCOUNT, or the number of rows in the data file if COMMITCOUNT is not specified. It is therefore recommended that the COMMITCOUNT option be specified to avoid transaction log overflow.</p> <p>This modifier is incompatible with INSERT_UPDATE mode, hierarchical tables, and the following modifiers: usedefaults, identitymissing, identityignore, generatedmissing, and generatedignore.</p> |

Table 22. Valid file type modifiers for the import utility: All file formats (continued)

| Modifier | Description |
|------------------|---|
| generatedignore | This modifier informs the import utility that data for all generated columns is present in the data file but should be ignored. This results in all values for the generated columns being generated by the utility. This modifier cannot be used with the generatedmissing modifier. |
| generatedmissing | If this modifier is specified, the utility assumes that the input data file contains no data for the generated columns (not even NULLs), and will therefore generate a value for each row. This modifier cannot be used with the generatedignore modifier. |
| identityignore | This modifier informs the import utility that data for the identity column is present in the data file but should be ignored. This results in all identity values being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with the identitymissing modifier. |
| identitymissing | If this modifier is specified, the utility assumes that the input data file contains no data for the identity column (not even NULLs), and will therefore generate a value for each row. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This modifier cannot be used with the identityignore modifier. |
| lobsinfile | <p><i>lob-path</i> specifies the path to the files containing LOB data.</p> <p>Each path contains at least one file that contains at least one LOB pointed to by a Lob Location Specifier (LLS) in the data file. The LLS is a string representation of the location of a LOB in a file stored in the LOB file path. The format of an LLS is <i>filename.ext.nnn.mmm/</i>, where <i>filename.ext</i> is the name of the file that contains the LOB, <i>nnn</i> is the offset in bytes of the LOB within the file, and <i>mmm</i> is the length of the LOB in bytes. For example, if the string <i>db2exp.001.123.456/</i> is stored in the data file, the LOB is located at offset 123 in the file <i>db2exp.001</i>, and is 456 bytes long.</p> <p>The LOBS FROM clause specifies where the LOB files are located when the "lobsinfile" modifier is used. The LOBS FROM clause will implicitly activate the LOBSINFILE behavior. The LOBS FROM clause conveys to the IMPORT utility the list of paths to search for the LOB files while importing the data.</p> <p>To indicate a null LOB, enter the size as -1. If the size is specified as 0, it is treated as a 0 length LOB. For null LOBS with length of -1, the offset and the file name are ignored. For example, the LLS of a null LOB might be <i>db2exp.001.7.-1/</i>.</p> |
| 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. |
| nodefaults | <p>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:</p> <ul style="list-style-type: none"> • If a default value can be specified for a column, the default value is loaded • If the column is nullable, and a default value cannot be specified for that column, a NULL is loaded • If the column is not nullable, and a default value cannot be specified, an error is returned, and the utility stops processing. |
| norowwarnings | Suppresses all warnings about rejected rows. |

File type modifiers for the import utility

Table 22. Valid file type modifiers for the import utility: All file formats (continued)

| Modifier | Description |
|--------------|--|
| seclabelchar | <p>Indicates that security labels in the input source file are in the string format for security label values rather than in the default encoded numeric format. IMPORT converts each security label into the internal format as it is loaded. If a string is not in the proper format the row is not loaded and a warning (SQLSTATE 01H53) is returned. If the string does not represent a valid security label that is part of the security policy protecting the table then the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3243W) is returned.</p> <p>This modifier cannot be specified if the seclabelname modifier is specified, otherwise the import fails and an error (SQLCODE SQL3525N) is returned.</p> |
| seclabelname | <p>Indicates that security labels in the input source file are indicated by their name rather than the default encoded numeric format. IMPORT will convert the name to the appropriate security label if it exists. If no security label exists with the indicated name for the security policy protecting the table the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3244W) is returned.</p> <p>This modifier cannot be specified if the seclabelchar modifier is specified, otherwise the import fails and an error (SQLCODE SQL3525N) is returned.
Note: If the file type is ASC, any spaces following the name of the security label will be interpreted as being part of the name. To avoid this use the striptblanks file type modifier to make sure the spaces are removed.</p> |
| usedefaults | <p>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:</p> <ul style="list-style-type: none"> • For DEL files: two adjacent column delimiters (",,") or two adjacent column delimiters separated by an arbitrary number of spaces (" , ") are specified for a column value. • For DEL/ASC/WSF files: A row that does not have enough columns, or is not long enough for the original specification. <p>Note: For ASC files, NULL column values are not considered explicitly missing, and a default will not be substituted for NULL column values. NULL column values are represented by all space characters for numeric, date, time, and /timestamp columns, or by using the NULL INDICATOR for a column of any type to indicate the column is NULL.</p> <p>Without this option, if a source column contains no data for a row instance, one of the following occurs:</p> <ul style="list-style-type: none"> • For DEL/ASC/WSF files: If the column is nullable, a NULL is loaded. If the column is not nullable, the utility rejects the row. |

Table 23. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL)

| Modifier | Description |
|-------------------------|--|
| codepage= <i>x</i> | <p><i>x</i> is an ASCII character string. The value is interpreted as the code page of the data in the output data set. Converts character data from this code page to the application code page during the import operation.</p> <p>The following rules apply:</p> <ul style="list-style-type: none"> • For pure DBCS (graphic) mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive. • 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. <p>Notes:</p> <ol style="list-style-type: none"> 1. The codepage modifier cannot be used with the lobsinfile modifier. 2. If data expansion occurs when the code page is converted from the application code page to the database code page, the data might be truncated and loss of data can occur. |
| dateformat=" <i>x</i> " | <p><i>x</i> is the format of the date in the source file.² Valid date elements are:</p> <p>YYYY - Year (four digits ranging from 0000 - 9999)
 M - Month (one or two digits ranging from 1 - 12)
 MM - Month (two digits ranging from 1 - 12;
 mutually exclusive with M)
 D - Day (one or two digits ranging from 1 - 31)
 DD - Day (two digits ranging from 1 - 31;
 mutually exclusive with D)
 DDD - Day of the year (three digits ranging
 from 001 - 366; mutually exclusive
 with other day or month elements)</p> <p>A default value of 1 is assigned for each element that is not specified. Some examples of date formats are:</p> <p>"D-M-YYYY"
 "MM.DD.YYYY"
 "YYYYDDD"</p> |
| implieddecimal | <p>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, <i>not</i> 12345.00.</p> |

File type modifiers for the import utility

Table 23. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

| Modifier | Description |
|----------------|--|
| timeformat="x" | <p>x is the format of the time in the source file.² Valid time elements are:</p> <ul style="list-style-type: none"> H - Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system) HH - Hour (two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system; mutually exclusive with H) M - Minute (one or two digits ranging from 0 - 59) MM - Minute (two digits ranging from 0 - 59; mutually exclusive with M) S - Second (one or two digits ranging from 0 - 59) SS - Second (two digits ranging from 0 - 59; mutually exclusive with S) SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86399; mutually exclusive with other time elements) TT - Meridian indicator (AM or PM) <p>A default value of 0 is assigned for each element that is not specified. Some examples of time formats are:</p> <pre> "HH:MM:SS" "HH.MM TT" "SSSSS" </pre> |

Table 23. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

| Modifier | Description |
|---------------------|---|
| timestampformat="x" | <p>x is the format of the time stamp in the source file.² Valid time stamp elements are:</p> <ul style="list-style-type: none"> YYYY - Year (four digits ranging from 0000 - 9999) M - Month (one or two digits ranging from 1 - 12) MM - Month (two digits ranging from 01 - 12; mutually exclusive with M and MMM) MMM - Month (three-letter case-insensitive abbreviation for the month name; mutually exclusive with M and MM) D - Day (one or two digits ranging from 1 - 31) DD - Day (two digits ranging from 1 - 31; mutually exclusive with D) DDD - Day of the year (three digits ranging from 001 - 366; mutually exclusive with other day or month elements) H - Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system) HH - Hour (two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system; mutually exclusive with H) M - Minute (one or two digits ranging from 0 - 59) MM - Minute (two digits ranging from 0 - 59; mutually exclusive with M, minute) S - Second (one or two digits ranging from 0 - 59) SS - Second (two digits ranging from 0 - 59; mutually exclusive with S) SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86399; mutually exclusive with other time elements) UUUUUU - Microsecond (6 digits ranging from 000000 - 999999; mutually exclusive with all other microsecond elements) UUUUU - Microsecond (5 digits ranging from 00000 - 99999, maps to range from 000000 - 999990; mutually exclusive with all other microsecond elements) UUUU - Microsecond (4 digits ranging from 0000 - 9999, maps to range from 000000 - 999900; mutually exclusive with all other microsecond elements) UUU - Microsecond (3 digits ranging from 000 - 999, maps to range from 000000 - 999000; mutually exclusive with all other microsecond elements) UU - Microsecond (2 digits ranging from 00 - 99, maps to range from 000000 - 990000; mutually exclusive with all other microsecond elements) U - Microsecond (1 digit ranging from 0 - 9, maps to range from 000000 - 900000; mutually exclusive with all other microsecond elements) TT - Meridian indicator (AM or PM) <p>A default value of 1 is assigned for unspecified YYYY, M, MM, D, DD, or DDD elements. A default value of 'Jan' is assigned to an unspecified MMM element. A default value of 0 is assigned for all other unspecified elements. Following is an example of a time stamp format:</p> <pre style="margin-left: 40px;">"YYYY/MM/DD HH:MM:SS.UUUUUU"</pre> <p>The valid values for the MMM element include: 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' and 'dec'. These values are case insensitive.</p> <p>The following example illustrates how to import data containing user defined date and time formats into a table called schedule:</p> <pre style="margin-left: 40px;">db2 import from delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule</pre> |

File type modifiers for the import utility

Table 23. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

| Modifier | Description |
|--------------------|---|
| usegraphiccodepage | <p>If usegraphiccodepage is given, the assumption is made that data being imported into graphic or double-byte character large object (DBCLOB) data fields is in the graphic code page. The rest of the data is assumed to be in the character code page. The graphic code page is associated with the character code page. IMPORT determines the character code page through either the codepage modifier, if it is specified, or through the code page of the application if the codepage modifier is not specified.</p> <p>This modifier should be used in conjunction with the delimited data file generated by drop table recovery only if the table being recovered has graphic data.</p> <p>Restrictions</p> <p>The usegraphi ccodepage modifier MUST NOT be specified with DEL files created by the EXPORT utility, as these files contain data encoded in only one code page. The usegraphi ccodepage modifier is also ignored by the double-byte character large objects (DBCLOBs) in files.</p> |
| xmlchar | <p>Specifies that XML documents are encoded in the character code page.</p> <p>This option is useful for processing XML documents that are encoded in the specified character code page but do not contain an encoding declaration.</p> <p>For each document, if a declaration tag exists and contains an encoding attribute, the encoding must match the character code page, otherwise the row containing the document will be rejected. Note that the character codepage is the value specified by the codepage file type modifier, or the application codepage if it is not specified. By default, either the documents are encoded in Unicode, or they contain a declaration tag with an encoding attribute.</p> |
| xmlgraphic | <p>Specifies that XML documents are encoded in the specified graphic code page.</p> <p>This option is useful for processing XML documents that are encoded in a specific graphic code page but do not contain an encoding declaration.</p> <p>For each document, if a declaration tag exists and contains an encoding attribute, the encoding must match the graphic code page, otherwise the row containing the document will be rejected. Note that the graphic code page is the graphic component of the value specified by the codepage file type modifier, or the graphic component of the application code page if it is not specified. By default, documents are either encoded in Unicode, or they contain a declaration tag with an encoding attribute.</p> <p>Note: If the xmlgraphic modifier is specified with the IMPORT command, the XML document to be imported must be encoded in the UTF-16 code page. Otherwise, the XML document may be rejected with a parsing error, or it may be imported into the table with data corruption.</p> |

Table 24. Valid file type modifiers for the import utility: ASC (non-delimited ASCII) file format

| Modifier | Description |
|----------------|---|
| nochecklengths | <p>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.</p> |

File type modifiers for the import utility

Table 24. Valid file type modifiers for the import utility: ASC (non-delimited ASCII) file format (continued)

| Modifier | Description |
|-----------------------|--|
| nullindchar= <i>x</i> | <p><i>x</i> is a single character. Changes the character denoting a null value to <i>x</i>. The default value of <i>x</i> is Y.³</p> <p>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.</p> |
| reclen= <i>x</i> | <p><i>x</i> is an integer with a maximum value of 32 767. <i>x</i> characters are read for each row, and a new-line character is not used to indicate the end of the row.</p> |
| striptblanks | <p>Truncates any trailing blank spaces when loading data into a variable-length field. If this option is not specified, blank spaces are kept.</p> <p>In the following example, <code>striptblanks</code> causes the import utility to truncate trailing blank spaces:</p> <pre>db2 import from myfile.asc of asc modified by striptblanks method 1 (1 10, 12 15) messages msgs.txt insert into staff</pre> <p>This option cannot be specified together with <code>striptnulls</code>. These are mutually exclusive options. This option replaces the obsolete <code>t</code> option, which is supported for back-level compatibility only.</p> |
| striptnulls | <p>Truncates any trailing NULLs (0x00 characters) when loading data into a variable-length field. If this option is not specified, NULLs are kept.</p> <p>This option cannot be specified together with <code>striptblanks</code>. These are mutually exclusive options. This option replaces the obsolete <code>padwithzero</code> option, which is supported for back-level compatibility only.</p> |

Table 25. Valid file type modifiers for the import utility: DEL (delimited ASCII) file format

| Modifier | Description |
|------------------|---|
| chardel <i>x</i> | <p><i>x</i> 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.³⁴ If you want to explicitly specify the double quotation mark as the character string delimiter, it should be specified as follows:</p> <pre>modified by chardel'"</pre> <p>The single quotation mark (') can also be specified as a character string delimiter. In the following example, <code>chardel''</code> causes the import utility to interpret any single quotation mark (') it encounters as a character string delimiter:</p> <pre>db2 "import from myfile.del of del modified by chardel'' method p (1, 4) insert into staff (id, years)"</pre> |
| coldel <i>x</i> | <p><i>x</i> 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.³⁴</p> <p>In the following example, <code>coldel;</code> causes the import utility to interpret any semicolon (;) it encounters as a column delimiter:</p> <pre>db2 import from myfile.del of del modified by coldel; messages msgs.txt insert into staff</pre> |
| decplusblank | <p>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.</p> |

File type modifiers for the import utility

Table 25. Valid file type modifiers for the import utility: DEL (delimited ASCII) file format (continued)

| Modifier | Description |
|-----------------|--|
| decptx | <p>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.³⁴</p> <p>In the following example, decpt; causes the import utility to interpret any semicolon (;) it encounters as a decimal point:</p> <pre>db2 "import from myfile.del of del modified by chardel' decpt; messages msgs.txt insert into staff"</pre> |
| delprioritychar | <p>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:</p> <pre>db2 import ... modified by delprioritychar ...</pre> <p>For example, given the following DEL data file:</p> <pre>"Smith, Joshua",4000,34.98<row delimiter> "Vincent,<row delimiter>, is a manager", 4005,44.37<row delimiter></pre> <p>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 <i>not</i> specified, there will be three rows in this data file, each delimited by a <row delimiter>.</p> |
| keepblanks | <p>Preserves the leading and trailing blanks in each field of type CHAR, VARCHAR, LONG VARCHAR, or CLOB. Without this option, all leading and trailing blanks that are not inside character delimiters are removed, and a NULL is inserted into the table for all blank fields.</p> |
| nochardel | <p>The import utility will assume all bytes found between the column delimiters to be part of the column's data. Character delimiters will be parsed as part of column data. This option should not be specified if the data was exported using DB2 (unless nochardel was specified at export time). It is provided to support vendor data files that do not have character delimiters. Improper usage might result in data loss or corruption.</p> <p>This option cannot be specified with chardelx, delprioritychar or nodoubledel. These are mutually exclusive options.</p> |
| nodoubledel | <p>Suppresses recognition of double character delimiters.</p> |

Table 26. Valid file type modifiers for the import utility: IXF file format

| Modifier | Description |
|----------|--|
| forcein | <p>Directs the utility to accept data despite code page mismatches, and to suppress translation between code pages.</p> <p>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.</p> |
| indexixf | <p>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 <i>insert-column</i> is specified.</p> |

Table 26. Valid file type modifiers for the import utility: IXF file format (continued)

| Modifier | Description |
|---------------------------------|---|
| <code>indexschema=schema</code> | Uses the specified <i>schema</i> for the index name during index creation. If <i>schema</i> is not specified (but the keyword <code>indexschema</code> is specified), uses the connection user ID. If the keyword is not specified, uses the schema in the IXF file. |
| <code>nochecklengths</code> | If <code>nochecklengths</code> 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. |
| <code>forcecreate</code> | Specifies that the table should be created with possible missing or limited information after returning SQL3311N during an import operation. |

Table 27. IMPORT behavior when using `codepage` and `usegraphiccodepage`

| <code>codepage=N</code> | <code>usegraphiccodepage</code> | IMPORT behavior |
|-------------------------|---------------------------------|---|
| Absent | Absent | All data in the file is assumed to be in the application code page. |
| Present | Absent | All data in the file is assumed to be in code page N.
Warning: Graphic data will be corrupted when imported into the database if N is a single-byte code page. |
| Absent | Present | Character data in the file is assumed to be in the application code page. Graphic data is assumed to be in the code page of the application graphic data.

If the application code page is single-byte, then all data is assumed to be in the application code page.

Warning: If the application code page is single-byte, graphic data will be corrupted when imported into the database, even if the database contains graphic columns. |
| Present | Present | Character data is assumed to be in code page N. Graphic data is assumed to be in the graphic code page of N.

If N is a single-byte or double-byte code page, then all data is assumed to be in code page N.

Warning: Graphic data will be corrupted when imported into the database if N is a single-byte code page. |

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. Double quotation marks around the date format string are mandatory. Field separators cannot contain any of the following: a-z, A-Z, and 0-9. The field separator should not be the same as the character delimiter or field delimiter in the DEL file format. A field separator is optional if the start and end positions of an element are unambiguous. Ambiguity can exist if (depending on the modifier) elements such as D, H, M, or S are used, because of the variable length of the entries.

File type modifiers for the import utility

For time stamp formats, care must be taken to avoid ambiguity between the month and the minute descriptors, since they both use the letter M. A month field must be adjacent to other date fields. A minute field must be adjacent to other time fields. Following are some ambiguous time stamp formats:

```
"M" (could be a month, or a minute)
"M:M" (Which is which?)
"M:YYYY:M" (Both are interpreted as month.)
"S:M:YYYY" (adjacent to both a time value and a date value)
```

In ambiguous cases, the utility will report an error message, and the operation will fail.

Following are some unambiguous time stamp formats:

```
"M:YYYY" (Month)
"S:M" (Minute)
"M:YYYY:S:M" (Month...Minute)
"M:H:YYYY:M:D" (Minute...Month)
```

Some characters, such as double quotation marks and back slashes, must be preceded by an escape character (for example, \).

3. 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. Delimiter restrictions for moving data lists restrictions that apply to the characters that can be used as delimiter overrides.
5. The following file type modifiers are not allowed when importing into a nickname:
 - indexixf
 - indexschema
 - dldelfiletype
 - nodefaults
 - usedefaults
 - no_type_idfiletype
 - generatedignore
 - generatedmissing
 - identityignore
 - identitymissing
 - lobsinfile
6. The WSF file format is not supported for XML columns.
7. The CREATE mode is not supported for XML columns.
8. All XML data must reside in XML files that are separate from the main data file. An XML Data Specifier (XDS) (or a NULL value) must exist for each XML column in the main data file.
9. XML documents are assumed to be in Unicode format or to contain a declaration tag that includes an encoding attribute, unless the XMLCHAR or XMLGRAPHIC file type modifier is specified.
10. Rows containing documents that are not well-formed will be rejected.

11. If the XMLVALIDATE option is specified, documents that successfully validate against their matching schema will be annotated with the schema information as they are inserted. Rows containing documents that fail to validate against their matching schema will be rejected. To successfully perform the validation, the privileges held by the user invoking the import must include at least one of the following:
- SYSADM or DBADM authority
 - USAGE privilege on the XML schema to be used in the validation

Related reference:

- “Delimiter restrictions for moving data” on page 826
- “db2Import API - Import data into a table, hierarchy, nickname or view” in *Administrative API Reference*
- “IMPORT ” on page 494

File type modifiers for the export utility

Table 28. Valid file type modifiers for the export utility: All file formats

| Modifier | Description |
|------------------|--|
| lobsinfile | <p><i>lob-path</i> specifies the path to the files containing LOB data.</p> <p>Each path contains at least one file that contains at least one LOB pointed to by a Lob Location Specifier (LLS) in the data file. The LLS is a string representation of the location of a LOB in a file stored in the LOB file path. The format of an LLS is <i>filename.ext.nnn.mmm/</i>, where <i>filename.ext</i> is the name of the file that contains the LOB, <i>nnn</i> is the offset in bytes of the LOB within the file, and <i>mmm</i> is the length of the LOB in bytes. For example, if the string <i>db2exp.001.123.456/</i> is stored in the data file, the LOB is located at offset 123 in the file <i>db2exp.001</i>, and is 456 bytes long.</p> <p>If you specify the “lobsinfile” modifier when using EXPORT, the LOB data is placed in the locations specified by the LOBS TO clause. Otherwise the LOB data is sent to the data file directory. The LOBS TO clause specifies one or more paths to directories in which the LOB files are to be stored. There will be at least one file per LOB path, and each file will contain at least one LOB. The LOBS TO or LOBFILE options will implicitly activate the LOBSINFILE behavior.</p> <p>To indicate a null LOB , enter the size as -1. If the size is specified as 0, it is treated as a 0 length LOB. For null LOBS with length of -1, the offset and the file name are ignored. For example, the LLS of a null LOB might be <i>db2exp.001.7.-1/</i>.</p> |
| xmlinsefiles | Each XQuery Data Model (QDM) instance is written to a separate file. By default, multiple values are concatenated together in the same file. |
| lobsinsefiles | Each LOB value is written to a separate file. By default, multiple values are concatenated together in the same file. |
| xmlnodeclaration | QDM instances are written without an XML declaration tag. By default, QDM instances are exported with an XML declaration tag at the beginning that includes an encoding attribute. |
| xmlchar | QDM instances are written in the character codepage. Note that the character codepage is the value specified by the codepage file type modifier, or the application codepage if it is not specified. By default, QDM instances are written out in Unicode. |
| xmlgraphic | If the <i>xmlgraphic</i> modifier is specified with the EXPORT command, the exported XML document will be encoded in the UTF-16 code page regardless of the application code page or the codepage file type modifier. |

Table 29. Valid file type modifiers for the export utility: DEL (delimited ASCII) file format (continued)

| Modifier | Description |
|---------------------|---|
| timestampformat="x" | <p>x is the format of the time stamp in the source file.⁴ Valid time stamp elements are:</p> <ul style="list-style-type: none"> YYYY - Year (four digits ranging from 0000 - 9999) M - Month (one or two digits ranging from 1 - 12) MM - Month (two digits ranging from 01 - 12; mutually exclusive with M and MMM) MMM - Month (three-letter case-insensitive abbreviation for the month name; mutually exclusive with M and MM) D - Day (one or two digits ranging from 1 - 31) DD - Day (two digits ranging from 1 - 31; mutually exclusive with D) DDD - Day of the year (three digits ranging from 001 - 366; mutually exclusive with other day or month elements) H - Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system) HH - Hour (two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system; mutually exclusive with H) M - Minute (one or two digits ranging from 0 - 59) MM - Minute (two digits ranging from 0 - 59; mutually exclusive with M, minute) S - Second (one or two digits ranging from 0 - 59) SS - Second (two digits ranging from 0 - 59; mutually exclusive with S) SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86399; mutually exclusive with other time elements) UUUUUU - Microsecond (6 digits ranging from 000000 - 999999; mutually exclusive with all other microsecond elements) UUUUU - Microsecond (5 digits ranging from 00000 - 99999, maps to range from 000000 - 999990; mutually exclusive with all other microsecond elements) UUUU - Microsecond (4 digits ranging from 0000 - 9999, maps to range from 000000 - 999900; mutually exclusive with all other microsecond elements) UUU - Microsecond (3 digits ranging from 000 - 999, maps to range from 000000 - 999000; mutually exclusive with all other microsecond elements) UU - Microsecond (2 digits ranging from 00 - 99, maps to range from 000000 - 990000; mutually exclusive with all other microsecond elements) U - Microsecond (1 digit ranging from 0 - 9, maps to range from 000000 - 900000; mutually exclusive with all other microsecond elements) TT - Meridian indicator (AM or PM) <p>Following is an example of a time stamp format:
"YYYY/MM/DD HH:MM:SS.UUUUUU"</p> <p>The MMM element will produce the following values: 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', and 'Dec'. 'Jan' is equal to month 1, and 'Dec' is equal to month 12.</p> <p>The following example illustrates how to export data containing user-defined time stamp formats from a table called 'schedule':</p> <pre>db2 export to delfile2 of del modified by timestampformat="yyyymm.dd hh:mm tt" select * from schedule</pre> |

File type modifiers for the export utility

Table 30. Valid file type modifiers for the export utility: IXF file format

| Modifier | Description |
|------------|--|
| codepage=x | <p>x is an ASCII character string. The value is interpreted as the code page of the data in the output data set. Converts character data from this code page to the application code page during the export operation.</p> <p>For pure DBCS (graphic), mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive. The codepage modifier cannot be used with the <code>lobsinfile</code> modifier.</p> |

Table 31. Valid file type modifiers for the export utility: WSF file format

| Modifier | Description |
|----------|--|
| 1 | Creates a WSF file that is compatible with Lotus 1-2-3 Release 1, or Lotus 1-2-3 Release 1a. ⁵ This is the default. |
| 2 | Creates a WSF file that is compatible with Lotus Symphony Release 1.0. ⁵ |
| 3 | Creates a WSF file that is compatible with Lotus 1-2-3 Version 2, or Lotus Symphony Release 1.1. ⁵ |
| 4 | Creates a WSF file containing DBCS characters. |

Notes:

- 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.
- Delimiter restrictions for moving data lists restrictions that apply to the characters that can be used as delimiter overrides.
- The export utility normally writes
 - date data in YYYYMMDD format
 - char(date) data in "YYYY-MM-DD" format
 - time data in "HH.MM.SS" format
 - time stamp data in "YYYY-MM-DD-HH. MM.SS. uuuuuu" format

Data contained in any datetime columns specified in the SELECT statement for the export operation will also be in these formats.

- For time stamp formats, care must be taken to avoid ambiguity between the month and the minute descriptors, since they both use the letter M. A month field must be adjacent to other date fields. A minute field must be adjacent to other time fields. Following are some ambiguous time stamp formats:

"M" (could be a month, or a minute)
 "M:M" (Which is which?)
 "M:YYYY:M" (Both are interpreted as month.)
 "S:M:YYYY" (adjacent to both a time value and a date value)

In ambiguous cases, the utility will report an error message, and the operation will fail.

Following are some unambiguous time stamp formats:

"M:YYYY" (Month)
 "S:M" (Minute)
 "M:YYYY:S:M" (Month...Minute)
 "M:H:YYYY:M:D" (Minute...Month)

- 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 can be specified.

6. The WSF file format is not supported for XML columns.
7. All QDM instances are written to XML files that are separate from the main data file, even if neither the "XMLFILE" nor the "XML TO" clause is specified. By default, XML files are written to the path of the exported data file. The default base name for XML files is the name of the exported data file with the ".xml" appended to it.
8. All QDM instances are written with an XML declaration at the beginning that includes an encoding attribute, unless the XMLNODEDECLARATION file type modifier is specified.
9. By default, all QDM instances are written in Unicode unless the XMLCHAR or XMLGRAPHIC file type modifier is specified.
10. The default path for XML data and LOB data is the path of the main data file. The default XML file base name is the main data file. The default LOB file base name is the main data file. For example, if the main data file is
/mypath/myfile.del

, the default path for XML data and LOB data is
/mypath"

, the default XML file base name is
myfile.del

, and the default LOB file base name is
myfile.del

.

The LOBSINFILE file type modifier must be specified in order to have LOB files generated.

11. The export utility appends a numeric identifier to each LOB file or XML file. The identifier is a 3 digit, 0 padded sequence value, starting at
.001

. After the 999th LOB file or XML file, the identifier will no longer be padded with zeroes (for example, the 1000th LOG file or XML file will have an extension of

.1000

. Following the numeric identifier is a three character type identifier representing the data type, either

.lob

or

.xml

. For example, a generated LOB file would have a name in the format
myfile.del.001.lob

, and a generated XML file would be have a name in the format
myfile.del.001.xml

.

12. It is possible to have the export utility export QDM instances that are not well-formed documents by specifying an XQuery. However, you will not be

File type modifiers for the export utility

able to import or load these exported documents directly into an XML column, since XML columns can only contain complete documents.

Related reference:

- “Delimiter restrictions for moving data” on page 826
- “db2Export API - Export data from a database” in *Administrative API Reference*
- “EXPORT ” on page 433

Delimiter restrictions for moving data

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 might 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 charde10x0C colde1X1e ...
```

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,

Delimiter restrictions for moving data

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.

Related reference:

- "File type modifiers for the export utility" on page 821
- "File type modifiers for the import utility" on page 810
- "File type modifiers for the load utility" on page 799

Delimiter restrictions for moving data

Appendix D. DB2 Database technical information

Overview of the DB2 technical information

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

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

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

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

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

Documentation feedback

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

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

Related concepts:

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

Related tasks:

- “Invoking command help from the command line processor” on page 326
- “Invoking message help from the command line processor” on page 326
- “Updating the DB2 Information Center installed on your computer or intranet server” on page 835

Related reference:

- “DB2 technical library in hardcopy or PDF format” on page 830

DB2 technical library in hardcopy or PDF format

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

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

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

Table 32. DB2 technical information

| Name | Form Number | Available in print |
|--|--------------------|---------------------------|
| <i>Administration Guide: Implementation</i> | SC10-4221 | Yes |
| <i>Administration Guide: Planning</i> | SC10-4223 | Yes |
| <i>Administrative API Reference</i> | SC10-4231 | Yes |
| <i>Administrative SQL Routines and Views</i> | SC10-4293 | No |
| <i>Call Level Interface Guide and Reference, Volume 1</i> | SC10-4224 | Yes |
| <i>Call Level Interface Guide and Reference, Volume 2</i> | SC10-4225 | Yes |
| <i>Command Reference</i> | SC10-4226 | No |
| <i>Data Movement Utilities Guide and Reference</i> | SC10-4227 | Yes |
| <i>Data Recovery and High Availability Guide and Reference</i> | SC10-4228 | Yes |
| <i>Developing ADO.NET and OLE DB Applications</i> | SC10-4230 | Yes |
| <i>Developing Embedded SQL Applications</i> | SC10-4232 | Yes |
| <i>Developing SQL and External Routines</i> | SC10-4373 | No |

Table 32. DB2 technical information (continued)

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

Table 33. DB2 Connect-specific technical information

| Name | Form Number | Available in print |
|--|-------------|--------------------|
| <i>DB2 Connect User's Guide</i> | SC10-4229 | Yes |
| <i>Quick Beginnings for DB2 Connect Personal Edition</i> | GC10-4244 | Yes |

Table 33. DB2 Connect-specific technical information (continued)

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

Table 34. WebSphere® Information Integration technical information

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

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

Related concepts:

- "Overview of the DB2 technical information" on page 829
- "About the Release Notes" in *Release notes*

Related tasks:

- "Ordering printed DB2 books" on page 832

Ordering printed DB2 books

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

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

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

Procedure:

To order printed DB2 books:

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

Related concepts:

- "Overview of the DB2 technical information" on page 829

Related reference:

- "DB2 technical library in hardcopy or PDF format" on page 830

Displaying SQL state help from the command line processor

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

Procedure:

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

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

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

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

Related tasks:

- "Invoking command help from the command line processor" on page 326
- "Invoking message help from the command line processor" on page 326

Accessing different versions of the DB2 Information Center

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

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

Related tasks:

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

Displaying topics in your preferred language in the DB2 Information Center

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

Procedure:

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

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

Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.

- To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

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

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

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

Related concepts:

- “Overview of the DB2 technical information” on page 829

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

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

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

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

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

Note: Updates are also available on CD. For details on how to configure your Information Center to install updates from CD, see the related links. If update packages are available, use the Update feature to download the packages. (The Update feature is only available in stand-alone mode.)

3. Stop the stand-alone Information Center, and restart the DB2 Information Center service on your computer.

Procedure:

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

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

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

```
<DB2 Information Center dir>\doc\bin\help_start.bat
```
 - On Linux:
 - a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the /opt/ibm/db2ic/V9 directory.

- b. Run the help_start script using the fully qualified path for the DB2 Information Center:

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

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

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

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

Note: The help_end batch file contains the commands required to safely terminate the processes that were started with the help_start batch file. Do not use Ctrl-C or any other method to terminate help_start.bat.

- On Linux, run the help_end script using the fully qualified path for the DB2 Information Center:

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

Note: The help_end script contains the commands required to safely terminate the processes that were started with the help_start script. Do not use any other method to terminate the help_start script.

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

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

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

Related concepts:

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

Related tasks:

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

DB2 tutorials

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

Before you begin:

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

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

DB2 tutorials:

To view the tutorial, click on the title.

Native XML data store

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

Visual Explain Tutorial

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

Related concepts:

- “Visual Explain overview” in *Administration Guide: Implementation*

DB2 troubleshooting information

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

DB2 documentation

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

DB2 Technical Support Web site

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

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

Related concepts:

- “Introduction to problem determination” in *Troubleshooting Guide*
- “Overview of the DB2 technical information” on page 829

Terms and Conditions

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

Personal use: You may reproduce these Publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not

distribute, display or make derivative work of these Publications, or any portion thereof, without the express consent of IBM.

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

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

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

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

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

Appendix E. Notices

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

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

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

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

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

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

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

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

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

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

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

Trademarks

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

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

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

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

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

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

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

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

Appendix F. Contacting IBM

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

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

Index

Special characters

! shell command 311

A

abnormal termination
 restart command 673
access path
 optimizing 702
action precompile/bind option 355, 583
ACTIVATE DATABASE command 330
ADD CONTACT command 333
ADD CONTACTGROUP command 335
Add Database Partition Server to an Instance command 178
ADD DBPARTITIONNUM command 336
ADD XMLSCHEMA DOCUMENT command
 syntax 339
admin configuration
 file 441
 network parameter values 756
 resetting to default 663
 sample 441
administration server
 configuration 441
 creating 13
 dropping 13
advisors
 db2advis 23
 Design Advisor 23
anyorder file type modifier 557
APPC (Advanced Program-to-Program Communication)
 node
 uncataloging 752
ARCHIVE LOG command 341
ASC import file type 494
ATTACH command 344
Audit Facility Administrator Tool command 29
Auto-start instance command 118
AUTOCONFIGURE command 346
Autostart DAS command 3

B

BACKUP DATABASE command 349
Backup Services APIs (XBSA) 349
Benchmark Tool command 34
binary files 286
binarynumerics file type modifier 557
BIND command
 syntax 355
Bind File Description Tool command 43
bindfile precompile option 583
binding
 errors 395
 implicitly created schema 355, 583

blocking precompile/bind option 355, 583

C

CALL statement
 run through the CLP 785
case sensitivity
 commands 321
 in naming conventions 797
CATALOG DATABASE command
 syntax 372
CATALOG DCS DATABASE command 375
CATALOG LDAP DATABASE command 377
CATALOG LDAP NODE command 381
CATALOG LOCAL NODE command 382
CATALOG NAMED PIPE NODE command 384
CATALOG ODBC DATA SOURCE command 386
CATALOG TCP/IP NODE command 387
cataloging
 databases 372
 host database 375
CCSIDG precompile/bind option 355, 583
CCSIDM precompile/bind option 355, 583
CCSIDS precompile/bind option 355, 583
CHANGE DATABASE COMMENT command 390
change database partition server configuration command 176
CHANGE ISOLATION LEVEL command 392
Change search path command 56
chardel file type modifier
 export 433
 import 494
 load 557
charsub precompile/bind option 355, 583
check backup command 57
check incremental restore image sequence command 63
CLI (call level interface)
 configuration 451
CLI/ODBC static package binding tool command 45
CLIPKG precompile/bind option 355
CLOSE statement
 run through the CLP 785
CLP (command line processor)
 command syntax 311
 quitting 618
 terminating 744
cnulreqd precompile/bind option 355, 583
code page file type modifier 557
code pages
 EXPORT command 433
 IMPORT command 494
coldel file type modifier
 export 433
 import 494
 load 557
collection precompile/bind option 355, 583
command
 db2drvmp 88
command line processor (CLP)
 accessing databases through 311
 accessing help 311
 batch mode 311
 command mode 311
 description 311
 interactive input mode 311
 invoking 311
 line continuation character 321
 options 312
 quitting 311, 618
 return codes 320
 shell command 311
 SQL statements 785
 terminating 311, 744
 using 321
command line processor invocation
 command 311
command syntax
 CLP commands 311
commands
 ACTIVATE DATABASE 330
 ADD CONTACT 333
 ADD CONTACTGROUP 335
 ADD DBPARTITIONNUM 336
 ARCHIVE LOG 341
 ATTACH 344
 AUTOCONFIGURE 346
 BACKUP DATABASE 349
 BIND 355
 CATALOG DATABASE 372
 CATALOG DCS DATABASE 375
 CATALOG LDAP DATABASE 377
 CATALOG LDAP NODE 381
 CATALOG LOCAL NODE 382
 CATALOG NAMED PIPE NODE 384
 CATALOG ODBC DATA SOURCE 386
 CATALOG TCP/IP NODE 387
 CHANGE DATABASE COMMENT 390
 CHANGE ISOLATION LEVEL 392
 CREATE DATABASE 395
 CREATE TOOLS CATALOG 408
 dasauto 3
 dasctrl 4
 dasdrop 5

commands (continued)

dasmigr 6
dasupt 8
db2 311
DB2 JDBC Package Binder
 Utility 138
DB2 SQLJ Profile Binder 252
DB2 SQLJ Translator 307
db2_deinstall 10
db2_install 11
db2_recon_aid 232
db2admin 13
db2adutil 15
db2advis 23
db2audit 29
db2batch 34
db2bfd 43
db2cap 45
db2cc 49
db2cfexp 51
db2cfimp 53
db2chgpath 56
db2ckbkp 57
db2ckmig 61
db2ckrst 63
db2cli 65
db2cmd 66
db2dart 67
db2daslevel 71
db2dclgn 72
db2diag 75
db2drdat 86
db2empfa 90
db2eva 91
db2evmon 93
db2evtbl 95
db2exmig 99
db2expln 100
db2extsec 105
db2flsn 106
db2fm 108
db2fs 110
db2gcf 111
db2gov 113
db2govlg 115
db2gpmap 116
db2hc 117
db2iauto 118
db2iclus 119
db2icrt 122
db2idrop 125
db2ilist 127
db2imigr 128
db2inidb 130
db2inspf 132
db2isetup 133
db2iupdt 135
db2jdbcbind 138
db2ldcfg 140
db2level 141
db2licm 142
db2listvolumes 144
db2logsforrfd 145
db2look 146
db2ls 155
db2move 157
db2mqsln 165

commands (continued)

db2mscs 169
db2mtrk 173
db2nchg 176
db2ncrt 178
db2ndrop 180
db2osconf 181
db2pd 184
db2pdcfg 223
db2perfc 226
db2perfi 228
db2perfr 229
db2rbind 230
db2relocatedb 235
db2sampl 242
db2set 245
db2setup 248
db2sql92 249
db2sqljbind 252
db2sqljcustomize 259
db2sqljprint 270
db2start 271
db2stop 272
db2support 273
db2swtch 278
db2sync 279
db2systray 280
db2tapemgr 282
db2tbst 285
db2trc 286
db2uiddl 290
db2undgp 291
db2unins 292
db2untag 294
db2xprt 295
DEACTIVATE DATABASE 411
DECOMPOSE XML
 DOCUMENT 413
DEREGISTER 415
DESCRIBE 417
DETACH 423
disable_MQFunctions 299
doce_deinstall 296
doce_install 297
DROP CONTACT 424
DROP CONTACTGROUP 425
DROP DATABASE 426
DROP DBPARTITIONNUM
 VERIFY 428
DROP TOOLS CATALOG 429
ECHO 431
EDIT 432
enable_MQFunctions 301
EXPORT 433
FORCE APPLICATION 439
GET ADMIN CONFIGURATION 441
GET ALERT CONFIGURATION 443
GET AUTHORIZATIONS 449
GET CLI CONFIGURATION 451
GET CONNECTION STATE 453
GET CONTACTGROUP 454
GET CONTACTGROUPS 455
GET CONTACTS 456
GET DATABASE
 CONFIGURATION 457
GET DATABASE MANAGER
 CONFIGURATION 463

commands (continued)

GET DATABASE MANAGER
 MONITOR SWITCHES 468
GET DESCRIPTION FOR HEALTH
 INDICATOR 471
GET HEALTH NOTIFICATION
 CONTACT LIST 473
GET HEALTH SNAPSHOT 474
GET INSTANCE 477
GET MONITOR SWITCHES 478
GET RECOMMENDATIONS 481
GET ROUTINE 485
GET SNAPSHOT 487
HELP 492
HISTORY 493
IMPORT 494
INITIALIZE TAPE 511
INSPECT 512
installFixPack 303
invoking help 326
LIST ACTIVE DATABASES 518
LIST APPLICATIONS 520
LIST COMMAND OPTIONS 522
LIST DATABASE DIRECTORY 523
LIST DATABASE PARTITION
 GROUPS 526
LIST DBPARTITIONNUMS 528
LIST DCS APPLICATIONS 529
LIST DCS DIRECTORY 531
LIST DRDA INDOUBT
 TRANSACTIONS 533
LIST HISTORY 535
LIST INDOUBT
 TRANSACTIONS 538
LIST NODE DIRECTORY 541
LIST ODBC DATA SOURCES 544
LIST PACKAGES/TABLES 545
LIST TABLESPACE
 CONTAINERS 548
LIST TABLESPACES 550
LIST UTILITIES 555
LOAD 557
LOAD QUERY 576
Microsoft Cluster Server 119
MIGRATE DATABASE 579
MQ Listener 165
PING 581
PRECOMPILE 583
PRUNE HISTORY/LOGFILE 607
PUT ROUTINE 609
QUERY CLIENT 611
QUIESCE 612
QUIESCE TABLESPACES FOR
 TABLE 615
QUIT 618
REBIND 619
RECONCILE 623
RECOVER DATABASE 627
redirecting output 321
REDISTRIBUTE DATABASE
 PARTITION GROUP 633
REFRESH LDAP 636
REGISTER 637
REORG INDEXES/TABLE 644
REORGCHK 654
RESET ADMIN
 CONFIGURATION 663

- commands (*continued*)
 - RESET ALERT
 - CONFIGURATION 665
 - RESET DATABASE
 - CONFIGURATION 667
 - RESET DATABASE MANAGER
 - CONFIGURATION 669
 - RESET MONITOR 671
 - RESTART DATABASE 673
 - RESTORE DATABASE 675
 - REWIND TAPE 690
 - ROLLFORWARD DATABASE 691
 - RUNCMD 701
 - RUNSTATS 702
 - SET CLIENT 716
 - SET RUNTIME DEGREE 719
 - SET TABLESPACE
 - CONTAINERS 721
 - SET TAPE POSITION 723
 - SET UTIL_IMPACT_PRIORITY 724
 - SET WRITE 726
 - setup 305
 - sqlj 307
 - START DATABASE MANAGER 728
 - START HADR 734
 - STOP DATABASE MANAGER 736
 - STOP HADR 739
 - TAKEOVER HADR 741
 - TERMINATE 744
 - UNCATALOG DATABASE 745
 - UNCATALOG DCS DATABASE 747
 - UNCATALOG LDAP
 - DATABASE 749
 - UNCATALOG LDAP NODE 751
 - UNCATALOG NODE 752
 - UNCATALOG ODBC DATA
 - SOURCE 753
 - UNQUIESCE 754
 - UPDATE ADMIN
 - CONFIGURATION 756
 - UPDATE ALERT
 - CONFIGURATION 758
 - UPDATE ALTERNATE SERVER FOR
 - DATABASE 762
 - UPDATE ALTERNATE SERVER FOR
 - LDAP DATABASE 764
 - UPDATE CLI
 - CONFIGURATION 766
 - UPDATE COMMAND OPTIONS 768
 - UPDATE CONTACT 770
 - UPDATE CONTACTGROUP 771
 - UPDATE DATABASE
 - CONFIGURATION 772
 - UPDATE DATABASE MANAGER
 - CONFIGURATION 775
 - UPDATE HEALTH NOTIFICATION
 - CONTACT LIST 777
 - UPDATE HISTORY FILE 778
 - UPDATE LDAP NODE 780
 - UPDATE MONITOR SWITCHES 782
 - COMPLETE XMLSCHEMA command
 - syntax 394
 - compound file type modifier 494
 - configurations
 - administration
 - resetting to default 663
 - sample 441
- configurations (*continued*)
 - CLI, sample 451
 - database
 - resetting to default 667
 - sample 457
 - updating 772
 - database manager, sample 463
 - Configure DB2 database for problem
 - determination behavior command 223
 - configure LDAP environment
 - command 140
 - connect precompile option 583
 - CONNECT statement
 - run through the CLP 785
 - connectivity configuration export tool
 - command 51
 - connectivity configuration import tool
 - command 53
 - contacting IBM 843
 - continuation character
 - command line processor (CLP) 321
 - Control Center
 - starting 49
 - Control DB2 instance command 111
 - CREATE DATABASE command
 - description 395
 - create instance command 122
 - create sample database command 242
 - CREATE TOOLS CATALOG
 - command 408
 - cursor stability (CS)
 - changing 392
- D**
 - DAS (DB2 Administration Server)
 - configuration 441
 - creating 13
 - dropping 13
 - dasauto command 3
 - dasrct command 4
 - dasdrop command 5
 - dasmigr command 6
 - dasupdt command 8
 - data
 - fragmentation, eliminating, by table
 - reorganization 644
 - data integrity
 - maintaining, with isolation levels 392
 - data skew
 - redistributing data in database
 - partition group 633
 - database analysis and reporting tool
 - command 67
 - database configuration
 - network parameter values 772
 - resetting to default 667
 - sample 457
 - updating 772
 - Database Connection Services (DCS)
 - directory
 - removing entries 747
 - database directories
 - changing comments 390
 - description 523
 - sample content 523
- database manager
 - accessing from command prompt 1
 - instances 477
 - monitor switches 468, 478
 - starting 728
 - statistics 487
 - stopping 736
 - system commands 1
 - database manager configuration
 - GET DATABASE MANAGER
 - CONFIGURATION command 463
 - sample file 463
 - database monitor
 - description 782
 - database movement tool command 157
 - database premigration tool command 61
 - database system monitor
 - GET DATABASE MANAGER
 - MONITOR SWITCHES
 - command 468
 - GET MONITOR SWITCHES
 - command 478
 - GET SNAPSHOT 487
 - RESET MONITOR command 671
 - UPDATE MONITOR SWITCHES
 - command 782
- databases
 - backup history file 607
 - cataloging 372
 - changing comments in directory 390
 - checking authorizations 449
 - deleting, ensuring recovery with log
 - files 426
 - dropping 426
 - exporting table to a file 433
 - home directory entry 523
 - importing file to table 494
 - indirect directory entry 523
 - information 487
 - loading file to table 557
 - migrating 579
 - monitor
 - resetting 671
 - recovering 691
 - remote directory entry 523
 - removing entries (uncataloging) 745
 - removing host DCS entries 747
 - reorganizing 654
 - restarting 673
 - restoring (rebuilding) 675
 - rollforward recovery 691
 - statistics 702
 - dateformat file type modifier 494, 557
 - datesiso file type modifier 433, 494, 557
 - DATETIME precompile/bind
 - option 355, 583
 - db2
 - CMD description 311
 - DB2 Administration Server (DAS)
 - creating 13
 - dropping 13
 - DB2 administration server command
 - creating 4
 - DB2 Administration Server command 13
 - db2 command 311
 - DB2 Connect
 - supported connections 375

DB2 database drive map command 88
 DB2 Fault Monitor command 108
 DB2 Governor command 113
 DB2 Governor Log Query command 115
 DB2 Index Advisor 23
 DB2 Information Center
 updating 835
 versions 833
 viewing in different languages 834
 DB2 Interactive CLI command 65
 DB2 JDBC Package Binder Utility
 command 138
 DB2 Profile Registry command 245
 DB2 SQLJ Profile Binder command 252
 DB2 SQLJ Profile Customizer
 command 259
 DB2 SQLJ Profile Printer command 270
 DB2 SQLJ Translator command 307
 DB2 Statistics and DDL Extraction Tool
 command 146
 db2_deinstall command 10
 db2_install command 11
 db2_recon_aid command 232
 db2admin command 13
 db2adutl command 15
 db2advis 23, 481
 db2audit command 29
 db2batch command 34
 db2bfd command 43
 db2cap command 45
 db2cc command 49
 db2cfexp command 51
 db2cfimp command 53
 db2chgpath command 56
 db2ckbkp command 57
 db2ckmig command 61
 db2ckrst command 63
 db2cli command 65
 db2cmd command 66
 db2dart command 67
 db2daslevel command 71
 db2dclgn declaration generator
 syntax 72
 db2diag command 75
 db2diag.log analysis tool command 75
 db2drdat command 86
 db2drvmp command 88
 db2empfa command 90
 db2eva command 91
 db2evmon command 93
 db2evtbl command 95
 db2exfmt tool 97
 db2exmig command 99
 db2expln command 100
 db2extsec command 105
 db2flsn command 106
 db2fm command 108
 db2fs command 110
 db2gcf command 111
 db2gov command 113
 db2govlg command 115
 db2gpmap command 116
 db2hc command 117
 db2iauto command 118
 db2iclus command 119
 db2icrt command 122
 db2idrop command 125
 db2ilist command 127
 db2imigr command 128
 db2inidb command 130
 db2inspf command 132
 db2isetup command 133
 db2iupdt command 135
 db2jdbcbind command 138
 db2ldcfg command 140
 db2level command 141
 db2licm command 142
 db2listvolumes command 144
 db2logsforrwd command 145
 db2look command 146
 db2ls command 155
 db2move command 157
 db2mqlsn command 165
 db2mscs command 169
 db2mtrk command 173
 db2nchg command 176
 db2ncrt command 178
 db2ndrop command 180
 DB2OPTIONS 312
 db2osconf command 181
 db2pd command 184
 db2pdcfg command 223
 db2perfc command 226
 db2perfi command 228
 db2perfr command 229
 db2rbind command 230
 db2relocatedb command 235
 db2rfpen command 240
 db2rspgn response file generator 241
 db2sampl command 242
 db2set command 245
 db2setup command 248
 db2sql92 command 249
 db2sqljbind command 252
 db2sqljcustomize command 259
 db2sqljprint command 270
 db2start command 271, 728
 db2stop command 272, 736
 db2support command 273
 db2swtch command 278
 db2sync command 279
 db2systay command 280
 db2tapemgr command 282
 db2tbst command 285
 db2trc command 286
 db2uiddl command 290
 db2undgp command 291
 db2unins command 292
 db2untag command 294
 db2xprrt command 295
 DCLGEN command
 db2dclgn declaration generator 72
 DEACTIVATE DATABASE
 command 411
 dec precompile/bind option 355, 583
 decdel precompile/bind option 355, 583
 Declaration Generator command 72
 DECLARE CURSOR statement
 run through the CLP 785
 DECOMPOSE XML DOCUMENT
 command
 description 413
 decplusblank file type modifier 433, 494,
 557
 decpt file type modifier 433, 494, 557
 default configuration
 admin, resetting to 663
 database, resetting to 667
 deferred_prepare precompile option 583
 degree precompile/bind option 355, 583
 delimiter restrictions
 moving data 826
 delprioritychar file type modifier 494,
 557
 Deregister command 415
 DESCRIBE command 417
 Design Advisor 23, 481
 DETACH command 423
 directories
 database
 changing comments 390
 Database Connection Services (DCS),
 uncataloging entries 747
 deleting entries 752
 node
 removing entries 752
 system database, removing 745
 uncataloging 745
 disable_MQFunctions command 299
 disconnect precompile option 583
 disconnecting
 command line processor front-end
 and back-end processes 744
 Display GUIDs for all disk volumes
 command 144
 dldel file type modifier 433, 494, 557
 doce_deinstall command 296
 doce_install command 297
 documentation 829, 830
 terms and conditions of use 837
 DRDA trace command 86
 DROP CONTACT command 424
 DROP CONTACTGROUP command 425
 DROP DATABASE command
 syntax 426
 Drop Database Partition Server from an
 Instance command 180
 DROP DBPARTITIONNUM VERIFY
 command 428
 DROP TOOLS CATALOG command 429
 dumpfile file type modifier 557
 dumping a trace to file 286
 DYNAMICRULES precompile/bind
 option
 BIND command 355
 PRECOMPILE command 583

E

ECHO command 431
 EDIT command 432
 Enable Multipage File Allocation
 command 90
 enable_MQFunctions command 301
 environment variables
 DB2OPTIONS 312
 error messages
 database configuration file 457
 dropping remote databases 426

error messages (*continued*)

- invalid checksum
 - database configuration file 667, 772
 - database manager configuration file 663
- Event Analyzer command 91
- Event Monitor Productivity Tool command 93
- exit codes (CLP)
 - list 320
- explain bind option 355, 583
- explain tables
 - formatting tool for data in 97
- explsnap precompile/bind option 355, 583
- EXPORT command 433
- export utility
 - file type modifiers 821
- exporting
 - database tables files 433
- exporting data
 - file type modifiers for 433

F

- fastparse file type modifier 557
- federated precompile/bind option 355, 583
- FETCH statement
 - run through the CLP 785
- file formats
 - exporting table to file 433
 - importing file to table 494
- file type modifiers
 - export utility 821
 - EXPORT utility 433
 - IMPORT command 494
 - import utility 810
 - LOAD command 557
 - load utility 799
- Find Log Sequence Number command 106
- First Steps 110
- FORCE APPLICATION command 439
- forcein file type modifier 494, 557
- Format inspect results command 132
- Format trap file command 295
- funcpath precompile/bind option 355, 583

G

- Generate Event Monitor Target Table Definitions command 95
- generatedignore file type modifier 494, 557
- generatedmissing file type modifier 494, 557
- generatedoverride file type modifier 557
- generic precompile/bind option 355, 583
- GET ADMIN CONFIGURATION command 441
- GET ALERT CONFIGURATION command 443
- GET AUTHORIZATIONS command 449

- GET CLI CONFIGURATION command 451
- GET CONNECTION STATE command 453
- GET CONTACTGROUP command 454
- GET CONTACTGROUPS command 455
- GET CONTACTS command 456
- GET DATABASE CONFIGURATION command 457
- GET DATABASE MANAGER CONFIGURATION command 463
- GET DATABASE MANAGER MONITOR SWITCHES command 468
- GET DESCRIPTION FOR HEALTH INDICATOR command 471
- Get distribution map command 116
- GET HEALTH NOTIFICATION CONTACT LIST command 473
- GET HEALTH SNAPSHOT command 474
- GET INSTANCE command 477
- GET MONITOR SWITCHES command 478
- GET RECOMMENDATIONS command 481
- GET ROUTINE command 485
- GET SNAPSHOT command 487
 - effect on UPDATE MONITOR SWITCHES 782
- Get Tablespace State command 285
- grant bind option 355
- grantgroup bind option 355
- grantuser bind option 355

H

- help
 - displaying 834
 - for commands 326
 - for messages 326
 - for SQL statements 833
- HELP command 492
- HISTORY command 493
- host systems
 - cataloging databases 375
 - connections supported by DB2 Connect 375
 - removing DCS catalog entries 747

I

- identityignore 494
- identityignore file type modifier 557
- identitymissing file type modifier 494, 557
- identityoverride file type modifier 557
- implicit connection 321
- implieddecimal file type modifier 494, 557
- IMPORT command 494
- import utility
 - file type modifiers 810
- importing data 494
- indexes
 - REORGCHK command 654

- indexes (*continued*)
 - statistics
 - RUNSTATS command 702
- indexfreespace file type modifier 557
- indexixf file type modifier 494
- indexschema file type modifier 494
- indoubt transaction field 538
- Information Center
 - updating 835
 - versions 833
 - viewing in different languages 834
- Initialize a Mirrored Database command 130
- INITIALIZE TAPE command 511
- insert precompile/bind option 355, 583
- INSPECT command 512
- install DB2 command 248
- Install DB2 command 305
- Install DB2 Information Center command 297
- Install DB2 product command 11
- installFixPack command 303
- invoking 326
 - command help 326
- IPX/SPX node
 - uncataloging 752
- isolation levels
 - CHANGE ISOLATION LEVEL command 392
- isolation precompile/bind option 355, 583

J

- JDBC Package Binder Utility command 138

K

- keepblanks file type modifier 494, 557

L

- LANGLEVEL precompile option 583
- SQL92E 583
- level precompile option 583
- License Management Tool command 142
- line continuation character
 - command line processor (CLP) 321
- LIST ACTIVE DATABASES command 518
- LIST APPLICATIONS command 520
- LIST COMMAND OPTIONS command 522
- LIST DATABASE DIRECTORY command 523
- LIST DATABASE PARTITION GROUPS command 526
- LIST DBPARTITIONNUMS command 528
- LIST DCS APPLICATIONS command 529
- LIST DCS DIRECTORY command 531
- LIST DRDA INDOUBT TRANSACTIONS command 533
- LIST HISTORY command 535

- LIST INDOUBT TRANSACTIONS
 - command 538
- List installed DB2 products and features
 - command 155
- List Instances command 127
- List Logs Required for Rollforward
 - Recovery command 145
- LIST NODE DIRECTORY command 541
- LIST ODBC DATA SOURCES
 - command 544
- LIST PACKAGES command 545
- LIST PACKAGES/TABLES
 - command 545
- LIST TABLES command 545
- LIST TABLESPACE CONTAINERS
 - command 548
- LIST TABLESPACES command 550
- LIST UTILITIES command 555
- LOAD command 557
- LOAD QUERY command 576
- load utility
 - file type modifiers 799
 - temporary files 557
- loading data
 - file to database table 557
 - file type modifiers for 557
- lobsinfile file type modifier 433, 494, 557
- locks
 - resetting maximum to default 667
- logs
 - listing during roll forward 691
- longerror precompile option 583

M

- Manage log files on tape command 282
- Memory Tracker command 173
- message help
 - invoking 326
- messages
 - accessing help 311
- messages precompile/bind option 355, 583
- Microsoft Cluster Server command 119
- MIGRATE DATABASE command 579
- Migrate explain tables command 99
- Migrate Instance command 128
- Migrate the DB2 Administration Server
 - command 6
- modifiers
 - file type
 - EXPORT command 433
 - export utility 821
 - IMPORT command 494
 - import utility 810
 - LOAD command 557
 - load utility 799
- Monitor and troubleshoot DB2 database
 - command 184
- monitoring
 - databases 468, 478
- moving data
 - between databases 494
 - delimiter restrictions 826
- MQ Listener command 165

N

- naming conventions
 - database manager objects 797
- NetBIOS
 - nodes
 - uncataloging 752
- no commit (NC) 392
- nochecklengths file type modifier 494, 557
- node directories, removing entries 752
- nodefaults file type modifier 494
- nodes
 - SOCKS 387
- nodoubledel file type modifier 433, 494, 557
- noeofchar file type modifier 494, 557
- noheader file type modifier 557
- NOLINEMACRO precompile option 583
- norowwarnings file type modifier 557
- notices 839
- notypeid file type modifier 494
- NULL string 321
- NULL value
 - command line processor
 - representation 321
- nullindchar file type modifier 494, 557

O

- Open DB2 Command Window
 - command 66
- OPEN statement
 - run through the CLP 785
- optimization
 - REORG INDEXES/TABLE
 - command 644
- optlevel precompile option 583
- ordering DB2 books 832
- output precompile option 583
- owner precompile/bind option 355, 583

P

- packages
 - recreating 619
- packages precompile option 583
- packeddecimal file type modifier 557
- pagefreespace file type modifier 557
- passwords
 - changing through CONNECT 785
 - changing with ATTACH
 - command 344
- performance
 - tuning
 - by reorganizing tables 644
 - REORGCHK command 654
- Performance Counters Registration Utility
 - command 228
- Performance Monitor Registration Tool
 - command 229
- phantom quiesce 615
- PING command 581
- PRECOMPILE command 583
- PREP command 583
- Prepare Unique Index Conversion to V5
 - Semantics command 290

- preprocessor precompile option 583
- printed books
 - ordering 832
- privileges
 - database
 - granted when creating 395
 - direct 449
 - indirect 449
 - report 449
- Problem Analysis and Environment
 - Collection Tool command 273
- problem determination
 - online information 837
 - tutorials 837
- PRUNE HISTORY/LOGFILE
 - command 607
- PUT ROUTINE command 609

Q

- qualifier precompile/bind option 355, 583
- QUERY CLIENT command 611
- queryopt precompile/bind option
 - BIND command 355
 - PRECOMPILE command 583
- QUIESCE command 612
- QUIESCE TABLESPACES FOR TABLE
 - command 615
- quiesce, phantom 615
- QUIT command 618

R

- read stability (RS)
 - changing 392
- Rebind all Packages command 230
- REBIND command 619
- reclen file type modifier 494
 - loading 557
- RECONCILE command
 - syntax 623
- Reconcile Multiple Tables command 232
- RECOVER DATABASE command 627
- recovery
 - database 675
 - with roll forward 691
 - without roll forward 675
- REDISTRIBUTE DATABASE PARTITION
 - GROUP command 633
- REFRESH LDAP command 636
- REGISTER command 637
- REGISTER XMLSCHEMA command
 - syntax 640
- REGISTER XSROBJECT command
 - syntax 642
- Release Container Tag command 294
- release precompile/bind option 355, 583
- Relocate Database command 235
- Remove a DB2 Administration Server
 - command 5
- Remove Instance command 125
- REORG TABLE command 644
- REORGCHK command 654
- repeatable read (RR)
 - changing 392

RESET ADMIN CONFIGURATION
 command 663
 RESET ALERT CONFIGURATION
 command 665
 RESET DATABASE CONFIGURATION
 command 667
 RESET DATABASE MANAGER
 CONFIGURATION command 669
 Reset Database Performance Values
 command 226
 RESET MONITOR command 671
 Reset rollforward pending state
 command 240
 response files
 generator
 db2rspgn 241
 RESTART DATABASE command 673
 RESTORE DATABASE command 675
 restoring
 earlier versions of DB2 databases 675
 RESTRICTIVE clause of CREATE
 DATABASE statement 395
 return codes
 command line processor (CLP) 320
 Revoke Execute Privilege command 291
 REWIND TAPE command 690
 ROLLFORWARD DATABASE
 command 691
 RUNCMD command 701
 RUNSTATS command
 syntax 702

S

schemas
 in new databases 395
 SELECT statement
 run through the CLP 785
 SELECT statements
 in EXPORT command 433
 SET CLIENT command 716
 Set permissions for DB2 objects
 command 105
 SET RUNTIME DEGREE command 719
 SET TABLESPACE CONTAINERS
 command 721
 SET TAPE POSITION command 723
 Set Up Windows Failover utility
 command 169
 SET UTIL_IMPACT_PRIORITY
 command 724
 SET WRITE command 726
 setup command 305
 show current DAS level command 71
 Show DB2 Service Level command 141
 SIGALRM signal
 starting database manager 728
 SIGINT signal
 starting database manager 728
 SOCKS node
 parameter 387
 special characters
 in commands 321
 SQL and XQuery Explain Command 100
 SQL statements
 accessing help 311
 displaying help 833

SQL statements (*continued*)
 using command line with 785
 SQL92-compliant SQL statement
 processor command 249
 sqlca precompile option 583
 sqlerror precompile/bind option 355,
 583
 sqlflag precompile option 583
 sqlj command 307
 SQLJ Profile Binder command 252
 SQLJ Translator command 307
 sqlrules precompile option 583
 sqlwarn precompile/bind option 355,
 583
 Start Control Center command 49
 START DATABASE MANAGER
 command 728
 Start DB2 command 271
 Start DB2 Synchronizer command 279
 Start DB2 system tray command 280
 START HADR command 734
 Start Health Center command 117
 Start Instance Creation Interface
 command 133
 starting
 DB2
 db2start command 271
 statistics
 database 702
 database manager 487
 reorganizing indexes 654
 REORGCHK 654
 STOP DATABASE MANAGER
 command 736
 Stop DB2 command 272
 STOP HADR command 739
 stopping
 DB2
 db2stop command 272
 storage
 physical 644
 strdel precompile/bind option 355, 583
 striptblanks file type modifier 494, 557
 striptnulls file type modifier 494, 557
 subtableconvert file type modifier 557
 Switch default DB2 copy command 278
 syncpoint precompile option 583
 syntax
 description 793
 for command line processor SQL
 statements 785
 system commands
 overview 1
 system database directory
 uncataloging 745

T

tables
 exporting to files 433
 importing files 494
 loading files to 557
 reorganization
 determining if required 654
 REORG INDEXES/TABLE
 command 644

tables (*continued*)
 statistics
 description 702
 TAKEOVER HADR command 741
 tape backup 349
 target precompile option 583
 TCP/IP
 node
 uncataloging 752
 temporary files
 LOAD command 557
 TERMINATE command 744
 termination
 abnormal 673
 command line processor back-end
 process 744
 normal 736
 terms and conditions
 use of publications 837
 text precompile/bind option 355, 583
 timeformat file type modifier 494, 557
 timestampformat file type modifier 494,
 557
 totalreespace file type modifier 557
 Trace command 286
 traces
 activating 286
 transform group precompile/bind
 option 355, 583
 troubleshooting
 online information 837
 tutorials 837
 true type font
 requirement for command line
 processor 321
 TSM archived images 15
 tutorials
 troubleshooting and problem
 determination 837
 Visual Explain 836

U

UNCATALOG DATABASE
 command 745
 UNCATALOG DCS DATABASE
 command 747
 UNCATALOG LDAP DATABASE
 command 749
 UNCATALOG LDAP NODE
 command 751
 UNCATALOG NODE command 752
 UNCATALOG ODBC DATA SOURCE
 command 753
 uncataloging
 database entries 745
 host DCS database entries 747
 system database directory 745
 uncommitted reads (UR)
 changing 392
 Uninstall DB2 Information Center
 command 296
 Uninstall DB2 products command 292
 Uninstall DB2 products or features
 command 10
 UNQUIESCE command 754

UPDATE ADMIN CONFIGURATION
command 756
UPDATE ALERT CONFIGURATION
command 758
UPDATE ALTERNATE SERVER FOR
DATABASE command 762
UPDATE ALTERNATE SERVER FOR
LDAP DATABASE command 764
UPDATE CLI CONFIGURATION
command 766
UPDATE COMMAND OPTIONS
command 768
UPDATE CONTACT command 770
UPDATE CONTACTGROUP
command 771
Update DAS command 8
UPDATE DATABASE CONFIGURATION
command 772
UPDATE DATABASE MANAGER
CONFIGURATION command 775
UPDATE HEALTH NOTIFICATION
CONTACT LIST command 777
UPDATE HISTORY FILE command 778
Update installed DB2 products
command 303
Update Instances command 135
UPDATE LDAP NODE command 780
UPDATE MONITOR SWITCHES
command 782
updates
DB2 Information Center 835
Information Center 835
usedefaults file type modifier 494, 557
user IDs
authorization 449
Utility for Kernel Parameter Values
command 181

V

validate precompile/bind option 355,
583
version precompile option 583
Visual Explain
tutorial 836

W

WCHARTYPE precompiler option
with Precompile command 583
wizards
db2advis 23
Design Advisor 23
Work with TSM Archived Images
command 15
workstations
remote
cataloging databases 372
removing catalog entries for
databases from 745
uncataloging from local
workstation 752

X

XBSA (Backup Services APIs) 349

XML schema repository
ADD XMLSCHEMA DOCUMENT
command 339
COMPLETE XMLSCHEMA
command 394
REGISTER XMLSCHEMA
command 640
REGISTER XSROBJECT
command 642

Z

zoned decimal file type modifier 557



Printed in USA

SC10-4226-00



Spine information:

IBM DB2 DB2 Version 9

Command Reference

